

RESEARCH ARTICLE

MTMO: an efficient network-centric algorithm for subtree counting and enumeration

Guanghui Li^{1,2}, Jiawei Luo^{1,*}, Zheng Xiao¹ and Cheng Liang¹

¹ College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

² School of Information Engineering, East China Jiaotong University, Nanchang 330013, China

* Correspondence: luojiawei@hnu.edu.cn

Received November 3, 2017; Revised December 14, 2017; Accepted December 25, 2017

Background: The frequency of small subtrees in biological, social, and other types of networks could shed light into the structure, function, and evolution of such networks. However, counting all possible subtrees of a prescribed size can be computationally expensive because of their potentially large number even in small, sparse networks. Moreover, most of the existing algorithms for subtree counting belong to the subtree-centric approaches, which search for a specific single subtree type at a time, potentially taking more time by searching again on the same network.

Methods: In this paper, we propose a network-centric algorithm (MTMO) to efficiently count k -size subtrees. Our algorithm is based on the enumeration of all connected sets of $k-1$ edges, incorporates a labeled rooted tree data structure in the enumeration process to reduce the number of isomorphism tests required, and uses an array-based indexing scheme to simplify the subtree counting method.

Results: The experiments on three representative undirected complex networks show that our algorithm is roughly an order of magnitude faster than existing subtree-centric approaches and base network-centric algorithm which does not use rooted tree, allowing for counting larger subtrees in larger networks than previously possible. We also show major differences between unicellular and multicellular organisms. In addition, our algorithm is applied to find network motifs based on pattern growth approach.

Conclusions: A network-centric algorithm which allows for a faster counting of non-induced subtrees is proposed. This enables us to count larger motif in larger networks than previously.

Keywords: complex network; evolutionary systems biology; network motif discovery; subtree counting; subtree isomorphism

Author summary: Network motifs are considered as simple building blocks of biological networks, which may help in network's function and evolution. However, discovering motifs could be computationally demanding because of the involvement of graph isomorphic detection. As any connected subgraph that is non-treelike can be generated by expanding a corresponding subtree through adding edges, this study proposes a network-centric algorithm (MTMO) to efficiently count tree-structured subgraphs. This enables us to discover larger network motif in larger networks than previously.

INTRODUCTION

The problem of counting the number of occurrences of a subgraph within a large graph is commonly termed subgraph counting. Related problems, such as network motif finding, relative graphlet frequency distance and graphlet degree distribution agreements, are all fundamental graph analysis methods to identify latent structure

in complex systems. They have applications in bioinformatics [1,2], chemoinformatics [3], online social network analysis [4], and many other areas. Recently, Liang *et al.* [5] presented a novel algorithm CoMoFinder to detect composite motifs in co-regulatory networks with two types of molecules including transcription factors and microRNAs. Consequently, we can exploit CoMoFinder to identify composite motifs in networks with different

types of nodes, such as drug-target interaction network [6], disease-ncRNA association network [7] and disease-microbe association network [8–10].

As any connected subgraph that is non-treelike can be generated by expanding a corresponding subtree through adding edges, this paper focuses on the problems of counting tree-structured subgraphs, which have been found to be very useful for revealing significant systemic differences among organisms [2,11]. Subtree mappings can be categorized into two, namely, induced and non-induced (see the Section of Preliminaries in Methods for definitions). The definition of non-induced subtree considers all possible occurrences of the subtree of interest within a network. Counting non-induced occurrences of subtrees is also desirable as real-world networks may include missing and spurious edges.

Counting the number of all possible non-induced subtrees is computationally demanding and a practical bottleneck for large networks. Omid *et al.* [12] introduced the MODA algorithm, which determines the frequency of a given subtree by searching for all possible mappings from the subtree into the input network. However, MODA is unable to scale to large networks with thousands of nodes, and the computation of very large subtrees remains exceedingly expensive. An alternative approach for estimating the number of non-induced subtrees and bounded treewidth subgraphs is based on the powerful color coding technique [13], which forms the basis for several recent serial and parallel implementations. Alon *et al.* [2] implemented color-coding subtree counting to demonstrate its applicability for finding large tree-structured motifs in biological networks. Zhao *et al.* [14,15] implemented distributed color-coding subtree counting for large graphs via both MPI and MapReduce, with applications in social network analysis. Recently, Slota and Madduri [16–18] proposed an algorithm called FASCIA, an efficient and scalable distributed implementation of subtree counting via color coding.

The aforementioned algorithms for counting non-induced occurrences of subtrees belong to the subtree-centric approaches, which search for one specific single subtree type at a time. Hence, subtree-centric methods have to be run once for each possible k -size subtrees, and all the different non-isomorphic k -size subtrees should be generated first. Although there are many motif-finding algorithms which exploit network-centric approaches, such as ESU [19] or Kavosh [20], there is no such study, to the best of our knowledge, that attempt to apply network-centric approaches to subtree counting problem. Network-centric approaches enumerate all subgraphs of a certain size, and then classify each enumerated subgraph into isomorphic classes. This means that isomorphism test will be required for each subgraph occurrence, while the number of actual non-isomorphic classes is much smaller,

particularly for tree topologies. For this reason, researchers recently devised several novel algorithms, such as QuateXelero [21] and FaSE [22], which incorporate a rooted tree data structure in the enumeration process to reduce the number of isomorphism tests required.

In 2011, Ferreira *et al.* [23] presented the first output-sensitive algorithm that enumerates all k -size subtrees in a graph G of size n in $O(sk)$ total time, where s is the number of these subtrees in G and scales as n^k . To our knowledge, the research is theoretical, with no experimental results for the purpose of comparison, and currently there is no improved algorithm for this enumeration problem [24].

In this paper, we derive a network-centric algorithm to count the number of all non-induced subtrees of a given size that occur in the input network. Our algorithm for counting non-induced occurrences of subtrees, called MTMO, consists of two subtasks, namely, enumeration and classification.

We provide significant improvements for the above two subtasks of counting non-induced subtrees. Based on the composition operation of an integer, we present a new algorithm for enumeration of the subtrees. Our proposed algorithm is much easier and faster to implement compared with the existing algorithms.

To avoid a large number of isomorphism tests, a rooted tree data structure, which represents the topology of the subtrees being enumerated, is searched during the actual enumeration. Each time a new vertex is added to the current enumerated subtree, a new ramification of the rooted tree is created, or we follow an already existing path. A path from the root node to any leaf in the created rooted tree corresponds to a different vertex connection pattern of a particular tree type. We have to compute a canonical labeling for each leaf to determine its subtree type, but we eliminate the need on all other occurrences that corresponds to the same path in the tree. Furthermore, we use an array-based indexing scheme to simplify the subtree counting method.

We evaluate our algorithm on three representative undirected complex networks. Experimental results show that we can obtain significant speedups, being roughly an order of magnitude faster than existing subtree-centric approaches and base network-centric algorithm which does not use rooted tree. Also, we show major differences between unicellular and multicellular organisms. In addition, our proposed algorithm is applied to identify network motif with yeast protein-protein interaction (PPI) data.

RESULTS

We use three representative networks to test the performance of our algorithm. One is a biological

network comprising protein-protein interaction network of the budding Yeast [25,26]. The other two are non-biological networks, namely, an electronic [27] and a dolphins social network [28,29]. For simplicity, all occurring self-loops are removed. The features of these networks are summarized in Table 1.

Unless otherwise specified, the platform used in these experiments is: Intel Xeon X5670 2.93 GHz 12 MB Cache 1,333 MHz with 48 GB main memory running under the Ubuntu 12.04 (amd64) operating system. The

source codes were compiled with the GNU gcc/g++ 4.6.3 compiler using the option “-O3”.

Effect of the labeled rooted tree

We compare MTMO with our base network-centric algorithm called NTMO, which does not use rooted tree and requires an isomorphism test for each subtree occurrence. We capture the runtime necessary to conduct a complete k -subtree counting on the original networks as

Table 1 Experimental datasets

Network	No. of vertices	No. of edges	Average degree	Maximum degree	Source
YeastPPI	2361	6646	5.630	64	[25,26]
Electronic	252	399	3.167	14	[27]
Dolphins	62	159	5.129	12	[28,29]

Table 2 Experimental results for MTMO vs. NTMO

Network	k	Classes	Occurrences	Processing times		Comparison vs. NTMO
				NTMO	MTMO	
YeastPPI	3	1	103504	0.08	0.00	NaN
YeastPPI	4	2	2445294	3.75	0.14	26.79×
YeastPPI	5	3	70997199	161.32	3.00	53.77×
YeastPPI	6	6	2332901289	6145.17	93.40	65.79×
YeastPPI	7	11	83309369433	261864.73	3701.55	70.74×
YeastPPI	8	23	3156730829473	—	128677.50	—
Average run time growth ratio				42.65	31.74	
Electronic	3	1	1161	0.00	0.00	NaN
Electronic	4	2	4625	0.01	0.00	NaN
Electronic	5	3	21627	0.06	0.00	NaN
Electronic	6	6	110370	0.31	0.01	31.00×
Electronic	7	11	591201	2.03	0.06	33.83×
Electronic	8	23	3255764	12.96	0.25	51.84×
Electronic	9	47	18267470	79.76	1.37	58.22×
Electronic	10	106	103983466	515.54	7.40	69.67×
Electronic	11	235	598953356	3247.79	39.83	81.54×
Electronic	12	551	3483503787	21075.99	233.29	90.34×
Average run time growth ratio				6.19	5.38	
Dolphins	3	1	923	0.00	0.00	NaN
Dolphins	4	2	6884	0.01	0.00	NaN
Dolphins	5	3	57434	0.14	0.00	NaN
Dolphins	6	6	506955	1.36	0.04	34.00×
Dolphins	7	11	4616856	15.27	0.28	54.54×
Dolphins	8	23	42742064	169.75	2.51	67.63×
Dolphins	9	47	397400342	1811.70	22.71	79.78×
Dolphins	10	106	3670658836	19618.11	219.86	89.23×
Dolphins	11	235	33379840446	183064.13	1790.84	102.22×
Dolphins	12	551	296840871353	—	17257.83	—
Average run time growth ratio				10.98	8.75	

“—” indicates that the experiment does not finish within 7 days.

we varied k from 3 to 12. The results are listed in Table 2.

In all cases, MTMO outperforms NTMO, with the speedup being roughly an order of magnitude faster. Furthermore, as the subtree size k increases, the speedup becomes greater as well. More precisely, the speedup is related to the ratio of number of subtree occurrences to the number of classes. For example, MTMO is up to 70 times faster when counting subtrees of size 7 in the Yeast network, but only 34 times faster in the electronic network. This finding is primarily because the number of subtree occurrences in Yeast is greater than that in the electronic network.

Comparison to other algorithms

MODA [12] is a recent open-source motif discovery tool that reports counts for non-induced subgraph occurrences. This tool requires Microsoft Visual Studio. To achieve a direct comparison, both MTMO and MODA are executed on the same computer with Intel Xeon X5670 2.93 GHz 12 MB Cache 1,333 MHz with 48 GB main memory running under the Windows 7 operating system. Considering that MODA is unable to scale to large networks with thousands of nodes, we use two other non-biological networks. As shown in the results listed in Table 3,

MTMO can attain substantial speedups compared with MODA. In addition, MTMO has less memory cost compared with MODA.

We further compare MTMO with FASCIA [16]. Slota and Madduri [16] reported the execution time for all possible subtrees of size 7 on the electronic network was approximately 22 s. According to Table 3, the execution time of MTMO for all possible subtrees of size 7 within the same network is under 1 s.

Comparison of PPI networks

In the following we download the PPI networks of four species, which consist of three unicellular organisms (*Saccharomyces cerevisiae*, *Escherichia coli*, and *Helicobacter pylori*) and a multicellular organism (*Caenorhabditis elegans*) from the DIP database (2014.10.1 updated) [30].

We calculate many topological statistics in these four networks, which are summarized in Table 4. All networks display small-world and scale-free properties. However, the clustering coefficient of all networks is exceptionally small. This is because the PPI networks of these species are far from complete.

We use the proposed algorithm to obtain normalized

Table 3 Experimental results for MTMO vs. MODA

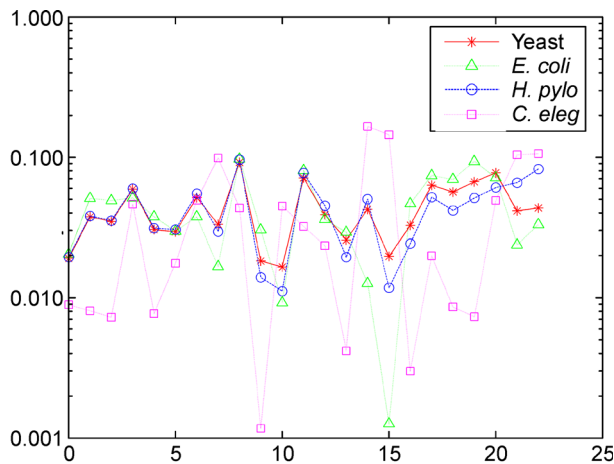
Network	k	Classes	Occurrences	Processing times		Comparison vs. MODA
				MODA	MTMO	
Electronic	3	1	1161	0.047	0.000	NaN
Electronic	4	2	4625	0.175	0.000	NaN
Electronic	5	3	21627	0.997	0.015	66.47×
Electronic	6	6	110370	5.692	0.062	91.81×
Electronic	7	11	591201	34.983	0.312	112.13×
Electronic	8	23	3255764	219.930	1.607	136.86×
Electronic	9	47	18267470	1416.216	9.079	155.99×
Electronic	10	106	103983466	8644.806	50.684	170.56×
Electronic	11	235	598953356	57515.849	287.587	199.99×
Electronic	12	551	3483503787	MEM	1673.030	—
Average run time growth ratio				5.840	5.290	
Dolphins	3	1	923	0.026	0.010	2.60×
Dolphins	4	2	6884	0.106	0.015	7.07×
Dolphins	5	3	57434	0.893	0.040	22.33×
Dolphins	6	6	506955	9.005	0.214	42.08×
Dolphins	7	11	4616856	84.713	1.862	45.50×
Dolphins	8	23	42742064	889.634	17.272	51.51×
Dolphins	9	47	397400342	10945.796	160.103	68.37×
Dolphins	10	106	3670658836	MEM	1532.602	—
Dolphins	11	235	33379840446	MEM	14884.071	—
Dolphins	12	551	296840871353	MEM	147292.440	—
Average run time growth ratio				9.130	7.330	

MEM indicates the out-of memory situation.

Table 4 Topological statistics in the PPI networks

Network	No. of vertices	No. of edges	Average degree	Clustering coefficient	Characteristic path length	Diameter	Power-law distribution $N=a k^{-r}$	
							α	r
<i>S. cere</i>	5137	22440	8.737	0.097	3.984	10	4014.5	1.707
<i>E. coli</i>	2955	11645	7.882	0.094	4.004	12	985.1	1.445
<i>H. pylo</i>	715	1364	3.815	0.017	4.135	9	352.2	1.651
<i>C. eleg</i>	2711	4042	2.982	0.022	4.837	14	601.8	1.607

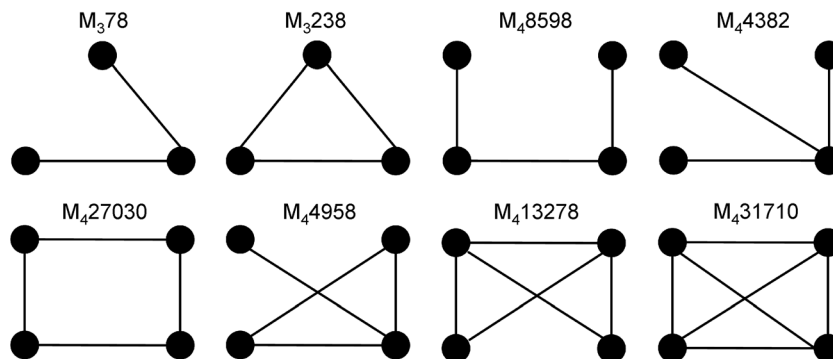
treelet distributions, that is, the number of non-induced occurrences of different tree topologies of size $k=8$ normalized by the total of all non-induced trees of size 8 for each PPI network. The corresponding treelet distributions are displayed in Figure 1. As can be seen, there are obvious differences between these three unicellular organisms and a multicellular organism, which is consistent with the one made by Alon *et al.* [2].

**Figure 1.** Normalized treelet distribution size 8 of the Yeast PPI network (Red), *E. coli* (Green), *H. pylo* (Blue) and *C. eleg* (Pink).

Network motif discovery based on pattern growth approach

The MODA algorithm introduced the concept of expansion trees based on pattern growth approach, which exploits the hierarchical structure of expansion trees by starting from k -subtrees and then extending these subtrees through adding edges to compute the frequency of each non-tree subgraph without using subgraph isomorphism. Therefore, to reduce the time of subtree search, we utilize MTMO to list all mappings of k -subtrees and then expanding these subtrees step by step until a complete graph. Here we apply our proposed algorithm to detect network motifs with yeast PPI data. The PPI data is downloaded from the Munich information center for protein sequences (MIPS) database [31] (marked as MIPS network), which contains 4,546 proteins and 12,319 interactions after filtering out the self-interactions and repeated edges. There are 2 non-isomorphic types for undirected 3-node subgraphs and 6 non-isomorphic types for undirected 4-node subgraphs. Figure 2 shows each undirected 3- and 4-node subgraph type, which is labeled as M_{ij} , where the subscript i represents the size of the subgraph, j denotes the code of the subgraph, which is a decimal number that transformed from the adjacency matrix of the subgraph.

Table 5 shows the statistical properties of each 3- and 4-node subgraph type in MIPS network. Since subgraphs with Z-score larger than 2.0 are identified as motifs, the five types of M_{3238} , M_{427030} , M_{44958} , M_{413278}

**Figure 2.** Shapes and labels for 3 and 4-node subgraphs in an undirected network.

and M_{431710} are detected as network motifs in MIPS network. Since MTMO is based on non-induced subgraphs as network motifs, we also extract induced subgraphs in the same network using ESU algorithm for comparison. As shown in Table 6, the five types of M_{3238} , M_{44382} , M_{44958} , M_{413278} and M_{431710} are network motifs.

As we can see the results from Table 5 and Table 6, four types of motifs are identical, but there is still one difference that the type of M_{427030} (a rectangle) is motif in our algorithm, whereas the type of M_{44382} (a star with three vertices) is motif in ESU. From the results of the existing algorithms, we found that motifs have high connectivity and treelike subgraphs are generally non-motifs. Therefore, non-induced subgraphs as network motifs may be more applicable, especially in incomplete PPI network.

DISCUSSION

On the basis of the detailed analyses of previous approaches, we have presented a network-centric algorithm which allows for a faster counting of non-induced subtrees. This enables to count larger subtrees in larger networks than previously possible, thereby facilitating future research on the extraction of combinatorial patterns. We also use the proposed algorithm to obtain treelet distributions for $k=8$ of the PPI networks and confirm previously reported differences between unicellular and multicellular organisms. Additionally, we apply our proposed algorithm to find network motifs based on pattern growth approach, which indicate that non-induced subgraphs may be more suitable for high quality network motif detection.

In this study, MTMO is applied to undirected networks. We find that applying MTMO to handle directed networks is also possible, as long as the labeled rooted tree data structure can represent the topological information of directed subtrees. Meanwhile, we will try to exploit multi-core parallel architectures, Graphic Processing Units

(GPUs), HAlign and Spark distributed computing system [34], and hadoop platform [35] to accelerate the MTMO algorithm.

METHODS

Preliminaries

In this paper, $G=(V, E)$ indicates an undirected graph, where V is a finite set of vertices, $|V|=n$ is the graph size, and E is the edge set of the graph that satisfies $E \subseteq V \times V$. A k -graph has size k . All vertices are assigned consecutive integers starting from 1.

We define a tree T that is a connected graph containing no cycle. Tree $T_s=(V_s, E_s)$ is a subtree of graph G if $V_s \subseteq V$ and $E_s \subseteq E \cap (V_s \times V_s)$. This subtree is non-induced when $\forall u, v \in V_s$: if $(u, v) \in E_s$ then $(u, v) \in E$ (in contrast to induced subtree, in which $(u, v) \in E$ if and only if $(u, v) \in E_s$). Therefore, non-induced subtree is decided by the edges that are selected, whereas induced subtree is decided by the vertices that are selected. For instance, in a triangle, there are 3 non-induced occurrences of pattern “ ∇ ” and one occurrence of pattern “ ∇ ”, whereas there is only one induced occurrence of pattern “ ∇ ” and no occurrence of pattern “ ∇ ”.

Two subtrees $T_1=(V_1, E_1)$ and $T_2=(V_2, E_2)$ are isomorphic if there exists a bijection between V_1 and V_2 such that two vertices are adjacent in T_1 if and only if their corresponding vertices in T_2 are adjacent.

For a particular subtree T_s , all subtrees isomorphic to T_s in the graph G are considered as occurrences of T_s . The frequency of a particular subtree in an input graph is the number of its occurrences in the graph.

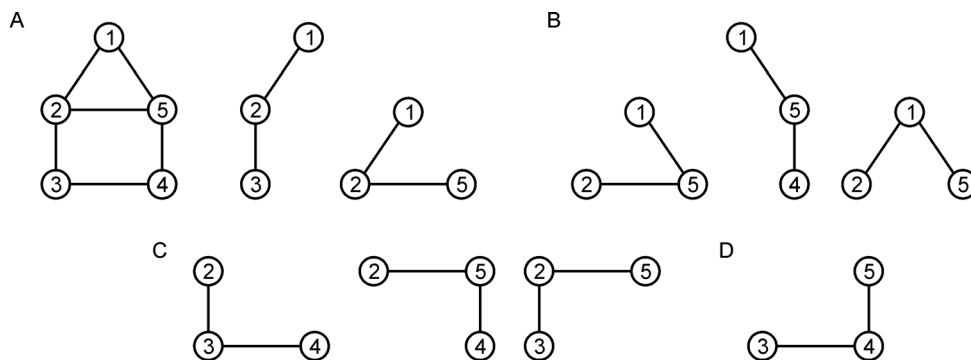
We will now define the problem we are attempting to solve in a more precise manner. *Definition 1 (Subtree Census Problem)*: given an input graph G and an integer k , determine the frequency of all non-induced k -subtrees of G . For example, the frequency is 9 for the particular 3-tree in the graph shown in Figure 3. Note that different occurrences can have overlaps in vertices or edges, that is,

Table 5 The statistical properties of MIPS network generated by our algorithm (non-induced subgraphs)

Code	Freq (Real)	Mean-freq (Random)	S-dev (Random)	Z-score
M_{378}	382545	382545.00	0.00	undefined
M_{3238}	10245	3256.45	131.34	53.21
M_{48598}	8002763	10047886.52	148466.57	-13.77
M_{44382}	13027193	13027193.00	0.00	undefined
M_{427030}	240724	64297.22	3166.95	55.71
M_{44958}	1732944	806416.51	43209.95	21.44
M_{413278}	234165	25512.19	2763.95	75.49
M_{431710}	15822	680.79	131.12	115.48

Table 6 The statistical properties of MIPS network generated by ESU (induced subgraphs)

Code	Freq (Real)	Mean-freq (Random)	S-dev (Random)	Z-score
M ₃ 78	97.1700%	99.9900%	0.00038	-73.3400
M ₃ 238	2.8300%	0.0152%	0.00038	73.3400
M ₄ 8598	27.0800%	38.6600%	0.00270	-42.8860
M ₄ 4382	66.1600%	60.9000%	0.00301	17.4990
M ₄ 27030	0.3055%	0.3849%	0.00019	-4.1973
M ₄ 4958	5.5766%	0.0624%	0.00158	34.8470
M ₄ 13278	0.7874%	0.0013%	4.22e-005	186.1800
M ₄ 31710	0.0895%	5.71e-006%	5.29e-007	1690.1000

**Figure 3.** Example graph G_1 and its 3-trees. (A) Graph G_1 . (B) 3-trees of G_1 including vertex 1. (C) 3-trees of $G_1 - \{1\}$ including vertex 2. (D) $G_1 - \{1, 2\} | 3$.

two occurrences of a subtree T , namely, T_1 and T_2 may share vertices. Actually, the vertex sets of T_1 and T_2 may be same. T_1 and T_2 are regarded as different occurrences of T , if they have at least one edge that they do not share.

Enumeration

Among the existing network-centric algorithms, Kavosh [20] is one of the best one. However, the algorithm focuses on induced occurrences. Note that an induced occurrence is determined by the selected vertices, while a non-induced occurrence is determined by the selected edges. In the present study, we propose an algorithm for enumerating non-induced subtrees that is similar to Kavosh in some aspects. In Kavosh, the subgraph is extended by neighboring vertices, whereas in our algorithm, the subtree is extended by neighboring edges. The process of the enumeration is explained in detail in the following discussion.

For enumerating all non-induced k -subtrees of a given graph G , all subtrees which include a particular vertex are first found. After that, this vertex is removed and the process is repeated for the remaining vertices.

For finding the k -subtrees in which a particular vertex participated, trees with maximum depth of k , rooted at this

vertex and based on neighborhood relationship are implicitly built. The children of each vertex include neighboring edges. A neighboring edge can be chosen as a child only if the corresponding neighboring vertex has not been included in the current enumerated subtree. Furthermore, all children in a particular tree must have numerical labels that are larger than that of the root of the tree.

The enumeration procedure uses the composition operation of an integer. A subtree of size k consists of $k-1$ edges. To extract k -subtrees, all possible compositions of integer $k-1$ must be taken into consideration. In mathematics, a composition of an integer $k-1$ is a sequence of positive integers with total sum $k-1$. Two sequences that differ in the order of their terms define different compositions of their summation. A composition can be expressed as k_1, k_2, \dots, k_m , where $k_1 + k_2 + \dots + k_m = k-1$. To find subtrees based on the composition, k_i edges are selected from the i -th level of the implicit tree to be edges of the subtrees ($i = 1, 2, \dots, m$). Meanwhile, any neighboring vertex cannot be selected repeatedly at each level of the implicit tree, to select an edge that will expand the subtree size by one. Finally, $k-1$ selected edges, which consist of k vertices, define a non-induced k -subtree within the graph G .

For a particular level i , $k_i < n_i$ is possible, where n_i is the number of edges present at level i . At level i , $C(n_i, k_i)$ different selections of edges should be considered. In this study, all combinations of edges are selected by using the “revolving door ordering” algorithm [34], which is the fastest algorithm used to generate combinations, and is a constant amortized time algorithm [35].

This step is described by an example on a given graph shown in Figure 4. For this graph, all 4-subtrees containing the vertex 1 are found, as illustrated in Figure 5.

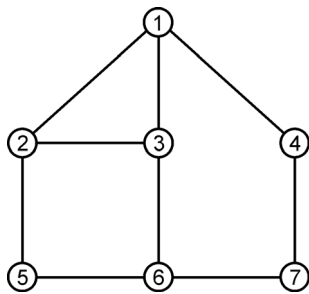


Figure 4. An instance of a network.

Classification

While performing the enumeration, we want a data structure to represent the topological information of the subtrees being enumerated. Hence, we construct a labeled rooted tree to perform a partial classification for enumerated subtrees. This data structure is similar to the quaternary tree in QuateXelero [21], but the set-up and usage of this data structure is slightly altered. Specifically, the labeled rooted tree in MTMO is built according to the parent information of newly added vertex, while the quaternary tree in QuateXelero is built according to the pattern of connections of newly added vertex. Compared to the quaternary tree, the labeled rooted tree in MTMO has less memory cost. In the remainder of this paper, we use the term “vertex” for a vertex in the input network and the term “node” for the nodes of the rooted tree to avoid confusion.

The enumeration process builds an implicit tree. Thus, there is one unique parent for each vertex that is added to the current enumerated subtree, except the first vertex. Hence, we use the labeled rooted tree to represent the parent information for each newly added vertex, as exemplified in Figure 6. For k -subtrees enumeration, the labeled rooted tree has the following properties:

- (i) The total length of the path from the root node to leaf is $k-1$.
- (ii) The level of the root node is assumed to be 0.

(iii) At i -th ($i = 0, 1, \dots, k-2$) level, each internal node has at most $i+1$ children.

(iv) At i -th ($i = 0, 1, \dots, k-1$) level, the number of nodes is at most $C(2i, i)/(i+1)$, which is the i -th Catalan number [36].

(v) Each edge is labeled with a number, indicating the parent information of the newly added vertex.

The labeled rooted tree can be searched along with the extension of subtree. That is, when the current enumerated subtree is extended by one vertex, the labeled rooted tree is likewise searched one level further using the parent information of the newly added vertex. An example of such a search is illustrated in Figure 7.

When the size of the current subtree reaches k , the current pointer of the labeled rooted tree will be moved to the appropriate leaf. All subtrees that reach the same leaf are isomorphic. However, two different leaves may correspond to the same subtree isomorphic class (Table 7). Consequently, the canonical labeling for each leaf must be computed to determine its subtree type only when the leaf is first created. Much work has already been done concerning tree isomorphism and we rely on AHU algorithm [37] for performing it in practice. For each rooted tree of size k , AHU algorithm produces a binary canonical labeling of length $2k$, which contains k ones and k zeros. An ordinary k -tree has one or two centers. Therefore, considering the center vertex as the root vertex, an ordinary tree can be converted into one or two rooted trees. So there are one or two canonical labelings for each class of tree isomorphism. All tree topologies for $k=5$ and the corresponding canonical labelings are listed in Figure 8. In order to simplify the subtree counting method, instead of searching a binary tree, we convert the canonical labeling into a decimal number, which corresponds to an array index. Thus, the frequency of each k -subtree is stored in a corresponding array element. However, the vast majority of the array elements are not used. For all different non-isomorphic class of a given size, the canonical labeling of star tree is minimal, while the canonical labeling of path tree is maximal; not much difference exists between the maximum and minimum. To conserve memory, we generate and store an array whose length is determined by the difference between the two values plus 1 (Table 8). Among these array elements, the first element stores the counts of star tree, while the last element stores the counts of path tree, and so on.

The pseudo code for our algorithm, which integrates the isomorphism tests into the enumeration process, is shown in Algorithm 1 (see Supplemental Materials). In this algorithm, the *Explore* procedure searches the labeled rooted tree while the subtree is extended. After being fully expanded, the subtree is classified. Finally, the actual frequencies are stored in an array.

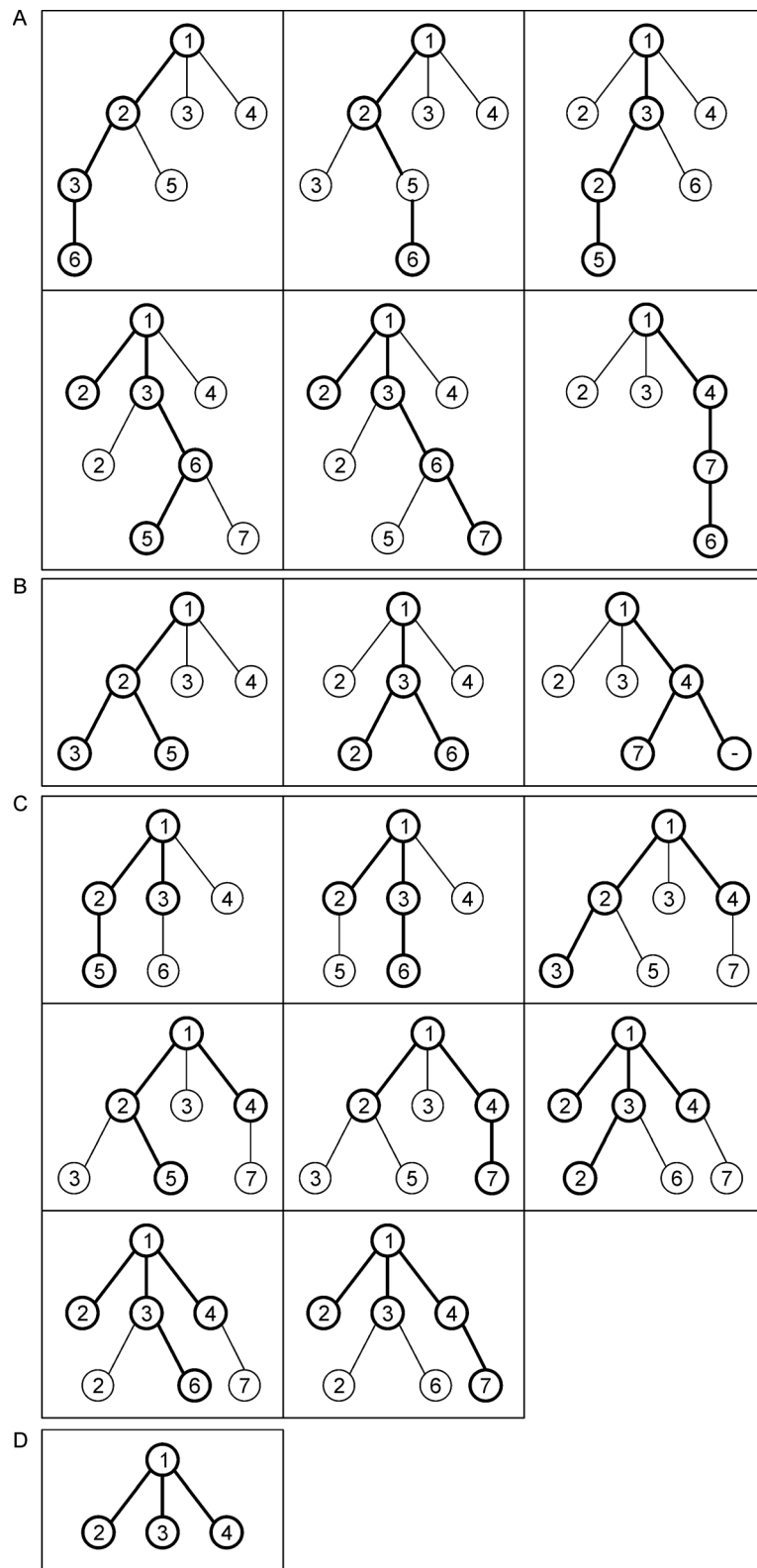


Figure 5. The implicit built trees rooted at vertex 1 of size 4 for network in Figure 4. (A) Trees built according to (1,1,1) pattern. All chosen vertices and edges are shown by specified circles in these figures. (B) Trees built according to (1, 2) pattern. (C) Trees built according to (2,1) pattern. (D) Tree built according to (3) pattern.

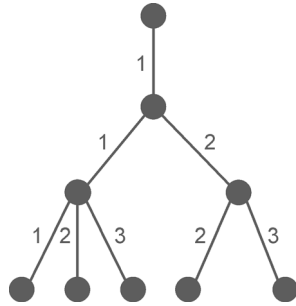


Figure 6. An example labeled rooted tree of enumerating 4-subtrees. In this figure, the number i indicates that the i -th vertex of current enumerated subtree is connected by newly added vertex.

It should be noted that in this algorithm (lines 10 to 12) the need to call the AHU algorithm is eliminated in many cases by using the labeled rooted tree. The time complexity of AHU algorithm is $O(k^2)$ in the worst case, while the maximal complexity of *Search* procedure is $O(k)$. As seen in Table 7, the number of leaves is less, which means that a great number of subtrees will reach the same leaf of the labeled rooted tree, and so calling the AHU algorithm will not be needed for them except for the first one. Consequently, this will remarkably reduce the computational time of subtree counting. In addition, the memory overhead that is used to construct the labeled rooted tree is negligible because the total number of nodes in the labeled rooted tree is less.

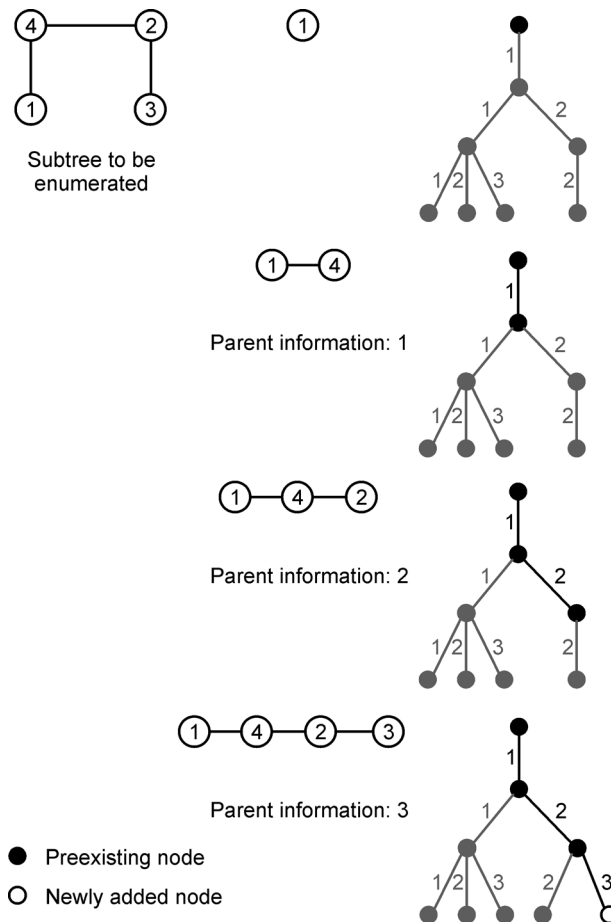


Figure 7. The steps of searching the rooted tree during enumerating a sample subtree. In this figure, the number i in the parent information indicates that the i -th vertex of current enumerated subtree is connected by newly added vertex.

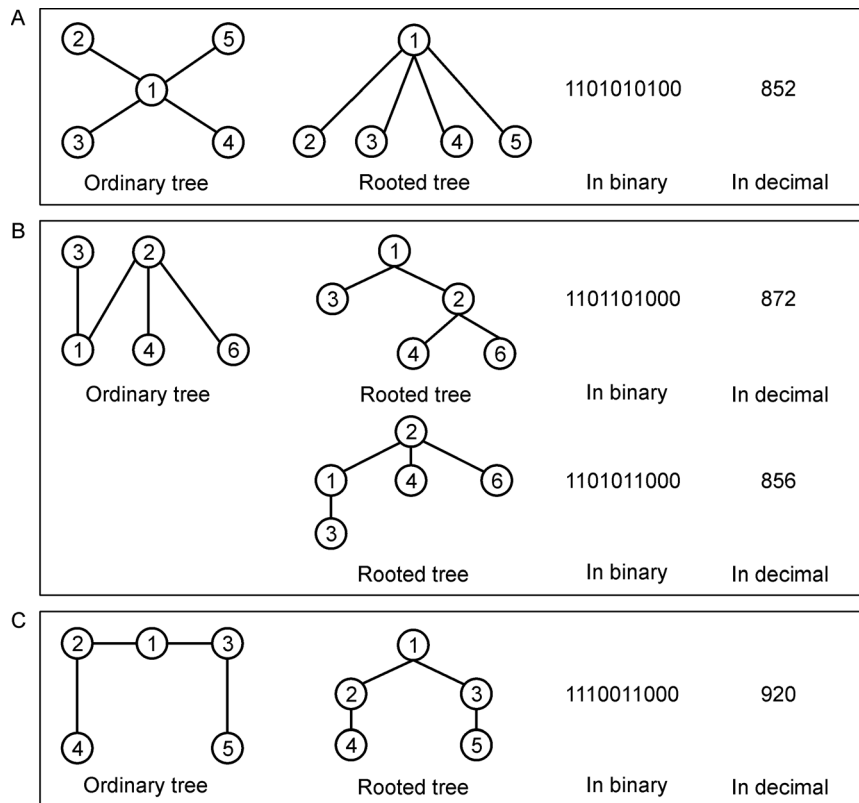


Figure 8. List of treelets for $k = 5$ with their canonical labelings. (A) The star tree with a center and 4 end-vertices. (B) The path tree with 3 vertices together with 2 end-vertices. (C) The path tree with 5 vertices.

Table 7 Number of each k -subtree non-isomorphic class, and the maximum number of corresponding labeled rooted tree leaves

k	3	4	5	6	7	8	9	10	11	12
Classes	1	2	3	6	11	23	47	106	235	551
Leaves	2	5	14	42	132	429	1430	4862	16796	58786

Table 8 The minimum and maximum of the canonical labeling for different k -subtrees, and the length of corresponding array

k	The minimum (in decimal)	The maximum (in decimal)	The length of array
3	52	52	1
4	212	216	5
5	852	920	69
6	3412	3696	285
7	13652	15472	1821
8	54612	61920	7309
9	218452	254432	35981
10	873812	1017792	143981
11	3495252	4130752	635501
12	13981012	16523136	2542125

Complexity Analysis of subtree enumeration

The process of the enumeration takes full advantage of the composition operation of an integer. Each positive integer m has 2^{m-1} distinct compositions [38]. For enumerating all k -subtrees in which a particular vertex participated, all distinct compositions of integer $k-1$ must be taken into consideration. According to the result of [38], there are 2^{k-2} compositions of $k-1$. Each composition need to select $k-1$ edges based on adjacent relationship. Meanwhile, considering that the degree of each vertex of G is not more than the hub degree (D), where D is the highest degree of G and D scales with n . Consequently, the worst complexity of subtree enumeration is $O(2^{k-2}nD^{k-1}) \approx O(2^{k-2}n^k)$. While the worst complexity of Ferreira's method is $O(kn^k)$ [23], which is slightly less than our method when $k \geq 5$. However, our proposed algorithm is much easier to implement compared with Ferreira's algorithm, which is only theoretical research.

SUPPLEMENTARY MATERIALS

The supplementary materials can be found online with this article at <https://doi.org/10.1007/s40484-018-0140-y>.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (No. 61572180) and Scientific and Technological Research Project of Education Department in Jiangxi Province (No. GJJ170383).

COMPLIANCE WITH ETHICS GUIDELINES

The authors Guanghui Li, Jiawei Luo, Zheng Xiao and Cheng Liang declare that they have no conflict of interests.

This article does not contain any studies with human or animal subjects performed by any of the authors.

REFERENCES

- Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. and Alon, U. (2002) Network motifs: simple building blocks of complex networks. *Science*, 298, 824–827
- Alon, N., Dao, P., Hajirasouliha, I., Hormozdiari, F. and Sahinalp, S. C. (2008) Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24, i241–i249
- Huan, J., Wang, W. and Prins, J. (2003) Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proc. Third IEEE Int'l Conf. on Data Mining*, pp. 549–552
- Kuramochi, M. and Karypis, G. (2001) Frequent subgraph discovery. In *Proc. First IEEE Int'l Conf. on Data Mining*, pp. 313–320
- Liang, C., Li, Y., Luo, J. and Zhang, Z. (2015) A novel motif-discovery algorithm to identify co-regulatory motifs in large transcription factor and microRNA co-regulatory networks in human. *Bioinformatics*, 31, 2348–2355
- Chen, X., Yan, C. C., Zhang, X., Zhang, X., Dai, F., Yin, J. and Zhang, Y. (2016) Drug-target interaction prediction: databases, web servers and computational models. *Brief. Bioinform.*, 17, 696–712
- Chen, X., Yan, C. C., Zhang, X. and You, Z. H. (2017) Long non-coding RNAs and complex diseases: from experimental results to computational models. *Brief. Bioinformatics*, 18, 558–576
- Chen, X., Huang, Y. A., You, Z. H., Yan, G. Y. and Wang, X. S. (2017) A novel approach based on KATZ measure to predict associations of human microbiota with non-infectious diseases. *Bioinformatics*, 33, 733–739
- Chen, X., Xie, D., Zhao, Q. and You, Z. H. (2017) MicroRNAs and complex diseases: from experimental results to computational models. *Brief. Bioinform.*, doi: 10.1093/bib/bbx130
- Chen, X., Yan, C. C., Zhang, X., You, Z. H., Deng, L., Liu, Y., Zhang, Y. and Dai, Q. (2016) WBSMDA: within and between score for MiRNA-disease association prediction. *Sci. Rep.*, 6, 21106
- Dao, P., Schönhuth, A., Hormozdiari, F., Hajirasouliha, I., Sahinalp, S. C. and Este, M. (2009) Quantifying systemic evolutionary changes by color coding confidence-scored PPI networks. In *9th Int'l Workshop on Algorithms in Bioinformatics*, pp. 37–48
- Omidi, S., Schreiber, F. and Masoudi-Nejad, A. (2009) MODA: an efficient algorithm for network motif discovery in biological networks. *Genes Genet. Syst.*, 84, 385–395
- Alon, N., Yuster, R. and Zwick, U. (1995) Color-coding. *J. Assoc. Comput. Mach.*, 42, 844–856
- Zhao, Z., Khan, M., Kumar, V. S. A. and Marathe, M. V. (2010) Subgraph enumeration in large social contact networks using parallel color coding and streaming. In *Proc. IEEE 39th Int'l Conf. on Parallel Processing*, pp. 594–603
- Zhao, Z., Wang, G., Butt, A. R., Khan, M., Kumar, V. S. A. and Marathe, M. V. (2012) SAHAD: subgraph analysis in massive networks using Hadoop. In *Proc. 26th Int'l. Parallel and Distributed Processing Symp.*, pp. 390–401
- Slota, G. M. and Madduri, K. (2013) Fast approximate subgraph counting and enumeration. In *Proc. IEEE 42nd Int'l Conf. on Parallel Processing*, pp. 210–219
- Slota, G. M. and Madduri, K. (2014) Complex network analysis using parallel approximate motif counting. In *Proc. 28th Int'l. Parallel and Distributed Processing Symp.*, pp. 405–414
- Slota, G. M. and Madduri, K. (2015) Parallel color-coding. *Parallel Comput.*, 47, 51–69
- Wernicke, S. (2006) Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3, 347–359
- Kashani, Z. R., Ahrabian, H., Elahi, E., Nowzari-Dalini, A., Ansari, E. S., Asadi, S., Mohammadi, S., Schreiber, F. and Masoudi-Nejad, A. (2009) Kavosh: a new algorithm for finding network motifs. *BMC Bioinformatics*, 10, 318
- Khakabimamaghani, S., Sharafuddin, I., Dichter, N., Koch, I. and Masoudi-Nejad, A. (2013) QuateXelero: an accelerated exact network motif detection algorithm. *PLoS One*, 8, e68073
- Paredes, P. and Ribeiro, P. (2013) Towards a faster network-centric

- subgraph census. In IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining, pp. 264–271
23. Ferreira, R., Grossi, R. and Rizzi, R. (2011) Output-sensitive listing of bounded-size trees in undirected graphs. In Proc. ESA'11, pp. 275–286
 24. Wasa, K. (2016) Enumeration of enumeration algorithms. arXiv:1605.05102
 25. Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., Zhang, J., Sun, S., Ling, L., Zhang, N., *et al.* (2003) Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res.*, 31, 2443–2450
 26. Batagelj, V. and Mrvar, A. (2006) Pajek Datasets. Available: <http://vlado.fmf.uni-lj.si/pub/networks/data/>
 27. ISCAS89 benchmark suite. <http://www.cbl.ncsu.edu/CBL Docs/iscas89.html>
 28. Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E. and Dawson, S. M. (2003) The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behav. Ecol. Sociobiol.*, 54, 396–405
 29. Newman, M. (2009) Network Data. Available: <http://www-personal.umich.edu/~mejn/netdata/>
 30. Xenarios, I., Salwinski, L., Duan, X. J., Higney, P., Kim, S. M. and Eisenberg, D. (2002) DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, 30, 303–305
 31. Mewes, H. W., Amid, C., Arnold, R., Frishman, D., Güldener, U., Mannhaupt, G., Münsterkötter, M., Pagel, P., Strack, N., Stümpflen, V., *et al.* (2004) MIPS: analysis and annotation of proteins from whole genomes. *Nucleic Acids Res.*, 32, D41–D44
 32. Wan, S. and Zou, Q. (2017) HAlign-II: efficient ultra-large multiple sequence alignment and phylogenetic tree reconstruction with distributed and parallel computing. *Algorithms Mol. Biol.*, 12, 25
 33. Zou, Q., Hu, Q., Guo, M. and Wang, G. (2015) HAlign: fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. *Bioinformatics*, 31, 2475–2481
 34. Stinson, D. (1999) *Combinatorial Algorithms: Generation, Enumeration, and Search*, pp. 48–52. Boca Raton: CRC Press
 35. Alamgir, Z. and Abbasi, S. (2007) Combinatorial algorithms for listing paths in minimal change order. In Proc. Fourth Conf. Combinatorial and Algorithmic Aspects of Networking, pp. 112–130
 36. Hilton, P. and Pedersen, J. (1991) Catalan numbers, their generalization, and their uses. *Math. Intell.*, 13, 64–75
 37. Aho, A., Hopcroft, J. and Ullman, J. (1974) *The Design and Analysis of Computer Algorithms*, pp 84–85. New Jersey: Addison-Wesley Publishing Co.
 38. Heubach, S. and Mansour, T. (2004) Compositions of n with parts in a set. *Congr. Numer.*, 168, 127–143