NeMo: Fast Count of Network Motifs

Michel M. Koskas, Gilles G. Grasseau, Etienne E. Birmelé, Sophie S. Schbath, Stephane S. Robin

NeMo: Fast Count of Network Motifs

Michel KOSKAS[1], Gilles GRASSEAU[2], Etienne BIRMELÉ[2], Sophie SCHBATH[3] and Stéphane ROBIN[1]

[1] UMR 518 AgroParisTech / INRA, 16 rue C. Bernard, 75005, Paris, France
{michel.koskas,stephane.robin}@agroparistech.fr
[2] Laboratoire Statistique & Génome, UMR CNRS 8071, INRA 1152, 523 place des Terrasses, 91000 Evry, France
{gilles.grasseau,etienne.birmele}@genopole.cnrs.fr
[3] INRA UR1077, Mathématique, Informatique et Génome, 78352 Jouy-en-Josas, France
sophie.schbath@jouy.inra.fr

Abstract *Networks is now the most popular way to describe interaction between biological objects. Studying network motifs is of particular interest in systems biology because these building blocks constitute functional units. We propose a tool to compute and statistically study the total number of occurrences of a given connected sub-graph, called topological motif, in a network. This tool relies on two very efficient algorithms to enumerate and/or count all the occurrences of a given topological motif in a given graph. Moreover, it implements approximate p-value computation in several probabilistic graph models extending some previous statistical results. The method is available through an R package named NeMo.*

Keywords Network motif, Motif count, Motif enumeration, R package.

1 Introduction

Networks is now the most popular way to describe interaction between biological objects. A classical approach to mathematically analyse networks from a static point of view aims at uncovering structures which can help in the understanding of the system. Global and local strategies can be considered and our work focuses on the latter. Studying substructures, called network motifs, is of particular interest in systems biology because these building blocks constitute functional units which combine to ensure cell regulatory processes ([1]). More than a pure structural role, motifs are conserved among species, suggesting that some local topologies are preferred from an evolutionary point of view ([2], [3], [4], [5]). Motifs of interest can be detected as exceptional motif, i.e. motifs with unexpected counts with respect to some reference model.

Several definition of network motifs can be considered. When vertices are colored (w.r.t. gene annotation for instance), a *coloured motif* is simply a multiset of colours whose occurrences correspond to connected sub-graphs with vertices having the relevant colors ([6], [7]). An algorithm to count the number of occurrences of such coloured motifs is proposed by [8] and their statistical significance is addressed in [9]. More commonly used, the *topological motifs* are connected sub-graphs with a fixed topology. These motifs can be directed or non-directed, depending on the nature of the biological network: protein-protein interaction (PPI) networks are in general non-directed whereas regulatory networks are directed. Globally over-represented topological motifs are often studied in regulatory networks because they reveal fundamental regulatory units ([10], [11], [1]). One may also be interested by detecting topological motifs locally over-represented ([12]). Further work has been done by collecting interest motifs among species and contracting vertices belonging to same ortholog groups, simplifying graphs and research of over-represented motifs onto a set of species ([30]).

Three steps are necessary to decide whether a motif $\mathbf{m}$ is over-represented in a given network G_{obs}.

- Counting the number of occurrences, denoted by $N_{\text{obs}} = N_{G_{\text{obs}}}(\mathbf{m})$, of the motif within the observed network;
- Defining some probabilistic random graph model which fits some desirable characteristics of the observed network G_{obs};
- Deriving the (approximate) p-value $\Pr\{N \geq N_{\text{obs}}\}$ under the model.

This global strategy has already been used in several papers cited above (e.g. [1]) and in [13]. In this paper, we focus on the first step.

Counting or enumerating network motifs is very computationally demanding and turns out to be a practical bottleneck for large graphs. Algorithms have been proposed by [1], [14] or [15]. Enumerating and counting motif occurrences are theoretically two very different problems. The first one consists in giving the list of occurrences of a motif. The second one consists in giving the cardinality of the enumeration. These two problems were up to now treated the same way: one counts the motif occurrences by enumerating them.

We propose two efficient algorithms: one dedicated to enumeration and the other to counting without complete enumeration. These two algorithms are exact. Thanks to their efficiency, they can handle large graphs (millions of vertices and motif of size few dozens of vertices in less than a minute on an average computer), avoiding commonly used sampling strategies ([16]), which only provide approximate counts. We also introduce the R package named NeMo dedicated to the detection of over-represented Network Motifs.

2 Methods

2.1 Motif Occurrence and Count

A topological motif (simply called motif in the sequel) $\mathbf{m}$ of size k ($k \geq 3$) is defined like a connected graph with k vertices.

Let G be a graph with n vertices labelled by $I = \{1, 2, \ldots, n\}$ and let denote by G_α the vertex-induced sub-graph of G on the set of vertices $\alpha = (i_1, \ldots, i_k)$, for some given $i_1, \ldots, i_k$ in I. We will say that an occurrence of motif $\mathbf{m}$ exists at position α in the graph G if and only if the vertex-induced sub-graph G_α is isomorphic to $\mathbf{m}$ or if G_α has a spanning subgraph isomorphic to $\mathbf{m}$. For instance, in a triangle, we will count 3 occurrences of the '$\vee$' motif and one of the '∇' motif.

Note that our definition of an occurrence differs from the occurrence of a vertex-induced sub-graph isomorphic to motif $\mathbf{m}$, which is often used in the literature ([10,14]). With this later definition, in a triangle, one would count no occurrence of the '$\vee$' motif and one occurrence of the '∇' motif. Our definition gives access to the total number of occurrences of the motif of interest. It is easy to switch between both way of counting, thanks to linear combinations ([17]). This way of counting occurrences is furthermore more robust to the fact that there might miss edges in the motif.

2.2 Enumerating Motif Occurrences

The first algorithm we propose is an exact enumeration (and counting) algorithm, based on a carefully organized backtracking. By convention, each adjacency of the graph (i.e. the pairs of indices of the connected vertices) is sorted. This algorithm relies on the following trick: the k vertices of the motif $\mathbf{m}$ are re-ordered, such that for all $1 \leq u \leq k$, the sub-motif $\mathbf{m}$ truncated to its first u vertices is connected (not strongly connected if the graph and motifs are oriented). This can be done for any connected motif.

This trick allows an early pruning of the exploration tree during backtracking. Indeed, an occurrence of $\mathbf{m}$ in the graph is a mapping φ from the set of vertices of $\mathbf{m}$ to the set of vertices of G such that for all u, v vertices of $\mathbf{m}$, if u and v are connected in $\mathbf{m}$ then $\varphi(u)$ and $\varphi(v)$ are connected in G. Therefore, if a vertex u of the motif is connected to several vertices with lower indices, say $u_1, \ldots, u_d$, the candidates in the graph to be $\varphi(u)$ will be at the intersection of the matching adjacencies of $\varphi(u_1), \ldots, \varphi(u_d)$. Intersecting sorted lists of integers can be very efficiently done thanks to a heap.

A simple backtracking using these features gives the following algorithm. In this algorithm, we call *a predecessor* of a vertex v of $\mathbf{m}$ the set of vertices of $\mathbf{m}$ connected to v but *whose indexes are lower than index of* v.

For both enumeration and counting algorithms, when searching the occurrences of a given motif, we avoid the search of automorphic occurrences by partitioning the vertices of the motif into classes such that two vertices v and v' of the motif $\mathbf{m}$ are in the same class if and only if there exists ψ an automorphism of $\mathbf{m}$ such

that $\psi(v) = v'$ or $\psi(v') = v$ (one of these conditions is sufficient to define the classes). When exploring the graph, we prune the backtrack by considering only mappings φ such that if v and v' are in the same class, $v < v' \Rightarrow \varphi(v) < \varphi(v')$.

ALGORITHM 1.

```
 Program SearchMotifs(Graph G, Motif m)
Input:  a graph G
        a Motif m
Output:  The list L of occurrences of m in G

Variables:  Partial list PL of vertices

Begin
L =<>
For  each  vertex i in G Do
    PL =< i >
    GoDeeper(G, m, PL, L)
EndFor
Return L
End

Procedure GoDeeper(Graph G, Motif m,
                   List of indices PL,
                   list of occurrences L)
Input:  a graph G
        the searched motif m
        The partial list of vertices
        (they are semantically the images
              of the first vertices of the
              motif in the graph G)
        The list of occurrences of m in G
Output:  Nothing (the goal
           of this procedure is to fill L)

Begin
If (PL.size() = M.size())
   Store PL in L
   return
EndIf
i = PL.size()
VI = i-th vertex of PL
Pr = list of predecessors
          of the i-th vertex in m
Pr2 = list of the vertices of PL (in G)
           corresponding to the vertices
           of Pr
Cand = intersection of the adjacencies
         of the vertices Pr2 greater than the
         image of the last vertex belonging to the same class as i
For each element e of Cand Do
   PL = PL + e
   GoDeeper(G, m, PL, L)
   PL = PL - e
End For each
End
```

Let us emphasize the following points.

1. The condition regarding the connectedness of any first vertices of the motif is crucial: it allows an early pruning of the exploration of the graph. A backtracking could take very significantly more time without this condition.
2. It is more efficient to index the vertices of the motif in order to maximize the degrees of the vertices to their predecessors in the motif.
3. The list of occurrences may be very huge even for small motifs (four vertices) and graphs (few thousands vertices). It usually will be impossible to store the occurrences in the memory of a machine and this list will usually be stored on a hard drive.

4. The intersection of a given number of lists is another crucial point of this algorithm. The adjacencies are considered as *sorted* lists of integers (the indexes of the connected vertices). It is very important to use a heap to perform efficiently this computation. The computation time is then $K \log h$ where h is the number of lists (the heap size) and K is the total number of elements of the union of these lists lower than or equal to the minimum of the maximums of the lists.
5. The graph might be oriented or not. In both cases the preceding algorithm holds. In the case of an oriented graph, the adjacencies are two arrays of sorted lists, the outgoing adjacencies and the incoming adjacencies for the graph and the motif. The adjacencies of whom one computes the intersection must have then the same direction in the motif and in the graph.
6. The efficiency is also increased by taking an order on the vertices of the network and computing the equivalence classes of $\mathbf{m}$. Stopping the backtracking whenever there exist u and v in the same equivalence class satisfying $u < v$ and $\phi(u) > \phi(v)$ ensures to visit each occurrence of the motif only once [18].

2.3 Counting Motif Occurrences

As any enumerating algorithm, the preceding algorithm may easily be changed into a counting algorithm: it suffices to replace the storage of the found occurrences with the incrementation of a counter. The second algorithm we propose allows counting without enumeration, which improve dramatically the performances, as shown in the next section.

Remind that we only consider motifs with at least three vertices which implies that at least one vertex has degree at least 2.

But in many cases, there is no need to explore the graph for the whole motif. Let V be the set of vertices of $\mathbf{m}$. Let S be the set of vertices of $\mathbf{m}$ connected to a single vertex of $\mathbf{m}$ and let $\mathbf{m}'$ be the motif $\mathbf{m}$ reduced to $V' = V \setminus S$. Let us denote $OutDeg(v')$ the degree of vertex $v' \in V'$ towards the vertices of S and $Deg(v)$ the degree of a vertex v in the motif. Let us call residual degree (denoted by $ResDeg()$) the degree of vertex $v' \in V'$ its number of connections to its neighbors in $\mathbf{m}'$.

Then let us suppose that we found an occurrence O of $\mathbf{m}'$ in G (we denote by φ' the mapping from V' to the vertices of G). Then the number of occurrences of $\mathbf{m}$ in G being a prolongation of O can be computed straightforwardly, taking care of possible crossings between the lists of neighbours.

Like in the enumeration algorithm, we avoid exploring automorphic occurrences.

We also remove from the graph its vertices of degree 1 (we obtain what we call here the reduced network, these vertices being unable to be any part of the reduced motif).

We hence modify the enumeration Algorithm 1 as follows:

ALGORITHM 2.

```
 Program CountMotifs(Graph G, Motif m)
Input:  a graph G
        a Motif m
Output:  The number of occurrences of m in G

Variables:  Partial list PL of vertices

Begin
N = 0
For  each  vertex v in G Do
    PL =< v >
    GoDeeper(G, m, PL, N)
EndFor
Return N
End

Procedure GoDeeper(Graph G, Motif m,
                   List of indices PL,
                   integer N)
Input:  a graph G
```

```
        the searched motif m
        The partial list of vertices
        The partial number N of occurrences
            of m in G
Output:  Nothing (the goal
           of this procedure is to compute N)

Begin
If (PL.size() = M.size())
   int PartialResult = ComputeCount(G, m, PL)
   Result = Result + PartialResult
   return
EndIf
i = PL.size()
VI = i-th vertex of PL
Pr = list of predecessors
          of the i-th vertex in m
Pr2 = list of the vertices of PL (in G)
           corresponding to the vertices
           of Pr
Cand = intersection of the adjacencies
         of the vertices Pr2
For each element e of Cand Do
   PL = PL + e
   GoDeeper(G, m, PL, N)
   PL = PL - e
End For each
End
```

Let us emphasize the following points.

1. This algorithm is faster than the preceding if and only if the motif has at least one vertex of degree 1, which is the most likely.
2. The counting of the motif $\mathbf{m}$ necessitates the enumeration of the motif $\mathbf{m}'$ (consisting of the vertices of $\mathbf{m}$ of degree at least 2). A further step would be to deduce the counting of $\mathbf{m}$ from the one of $\mathbf{m}'$, thanks to prior computations on the graph G.
3. This counting algorithm is very close to an enumeration algorithm. Let us illustrate this point through an example. Let us suppose the graph is composed of 9 vertices, the first three connected to each other, the three next connected to the vertex 1 and the three last ones being connected to vertex 2. Let us suppose that the wanted motif consists of 5 vertices, the three one connected to each other, the fourth one being connected to vertex 1 and the fifth one being connected to vertex 2. Then the reduced graph and motif are both triangles. Counting the occurrences is made as explained above but we could enumerate the occurrences of the full motif in the full graph with a writing like $\{1, 2, 3, [4, 5, 6], [7, 8, 9]\}$, brackets meaning one can take any vertex among theses ones.
4. The worst case for both algorithms is the case of cycles: both of these algorithms are relatively slow on enumerating or counting them. For example, it takes on an average computer up to 2.98 seconds to count or enumerate the squares in the PPI network of *D. melanogaster* (7068 vertices, 22532 edges). Into the same graph, counting or enumerating 5 -cycles takes more than 1 minute and 20 seconds (1 minute and 21.968 seconds).

Complexity: When the motif $\mathbf{m}$ is fixed, all backtracking algorithms have a polynomial complexity. When the degrees in the graph are bounded (it is usually the case in PPI graphs) then the complexity is bounded by a linear function in the number of edges (or arcs) of the graph. If the motif changes, the problem is known to be NP-hard in general and NP-complete for some particular families of motifs (the cliques for instance). The present algorithms have the same asymptotical complexities than any backtracking algorithm.

2.4 Results

We compared our algorithms with one of the most popular motif detection tool: FANMOD ([14]). Up to now, this software is the most efficient to count and enumerate all the present motifs in a network and

is generally used as a time reference to compare with other algorithms. Remind that FANMOD counts the number of vertex-induced occurrences, whereas we count the total number of occurrences.

There is another important difference between the general strategies of FANMOD and NeMo. FANMOD performs a systematic exploration of the network for sub-graphs of a given size k. This produces the list of occurrences only for motifs that are actually present in the network. Our algorithms deal with one motif at a time. They hence need to be run once for each possible motif of size k (called 'hereafter k-motifs'). We therefore need to first generate the list of all possible k-motifs, which is a CPU time consuming task as the number of operations increases as $\sim \exp(k^2)$. Nevertheless, for a given motif size k, this can be done once for all, and stored to be reused for multiple network analysis.

The performances are presented here for two PPI undirected networks.

– *E. coli* PPI network (424 vertices and 519 edges with 1.22 mean degree) is a small network often found in the literature dealing with the motif detection algorithm performances (see Table 1).
– *S. cerevisae* PPI network (4958 vertices and 17224 edges with 3.4 mean degree) offers a more dense network than the previous one (see Table 2).

The three codes were compiled with the same compiler options (`g++-4.4.3` with `-O3` optimization option) and run on the same computer (Intel Core i7 2.8 GHz with 2.9 Go memory, Ubuntu 10.04 LTS)

Motif size k	Number of motifs	Generation of the motif list (sec)	FANMOD (sec)	Algorithm 1 (sec)	Algorithm 2 (sec)
5	21	0	3	0.4	0
6	112	1.7	74	11	0.1
7	853	48	1 490	163	2
8	11 117	23 455	26 238	3 420	14

Table 1. Performances for the count of all k-motifs in the *E. coli* PPI network (undirected).

Motif size k	Number of motifs	Generation of the motif list (sec)	FANMOD (sec)	Algorithm 1 (sec)	Algorithm 2 (sec)
4	6	0	14	2.8	1.7
5	21	0	1 584	166	11
6	112	1.7	107 374	12 732	538

Table 2. Computation times to count all k-motifs in the *S. cerevisae* PPI network.

Algorithm 1 is 5 to 9 times faster than FANMOD. This gain is explained by the use of (at most) linear complexity operations in the internal backtracking procedure and by the constraint mechanism to avoid exploring the automorphisms of the same motif.

A drastic speed-up is obtained with Algorithm 2: from 750 to 1800 times for the *E. coli* PPI network and around 150-200 concerning the *S. cerevisae* PPI network. These factors are obtained thanks to the motif reduction which implies two types of gains: first the backtracking is pruned earlier (each step avoided gains a factor in computation time) and secondly the motifs are counted by groups (the same terminal case of the backtracking gives numerous counted occurrences without any further exploration, which also divides the computation time by the average number of counted occurrences at each terminal case).

3 Conclusion

3.1 The NeMo Package

We implemented the counting and enumerating algorithms in the R package NeMo which is dedicated to the detection of network motifs with unexpected count in an observed network. For this purpose, the package allows to assess the significance of observed counts with respect to three different random graph models:

– Erdös-Rényi ([19]), which is the simplest random graph model accounting only for the network density;
– Expected Degree Distribution ([20,21]), which fits the observed degree distribution, in average;

– MixNet ([22], also known as Stochastic Block-Model ([23]), which accounts for an arbitrary heterogeneity structure in the network.

The popular Fixed Degree Distribution (see e.g. [10]) is not among the proposed models, as it does not retrieve correctly the variability of the count of star-like motifs (see [13]).

For each of these models, the package provides the exact mean and variance of the count. A p-value based on the compound Poisson approximation, proposed in [13], is computed. We implemented the efficient calculation of the tail of this distribution proposed by [24]. NeMo can both handle directed and undirected networks.

As NeMo can both count and enumerate motif occurrences, it can be used in a 2 step strategy. Exceptional motifs among all k-motifs can be efficiently detected at first using the counting algorithm (Algorithm 2). Then, a deeper analysis can be carried for each exceptional motif, by looking at their location in the network using the enumerating algorithm (Algorithm 1).

The NeMo package will be soon available on the CRAN repository.

3.2 Discussion

We proposed an efficient algorithm to count topological motif occurrences in biological networks that outperforms other existing tools. The dramatic gain in the computation time provided by these algorithms allows us to use motif counts intensively for network analysis. As a first application, such efficient algorithms make possible the comparison of a large number of nerworks, based on motif counts ([25], [26], [27], [28]). Moreover, they allow to study the apparition of new motifs under complex evolutionary graph models. [29] proposed an Approximate Bayesian Computation strategy to infer such a model, using motif counts as summary statistics. The computation time dedicated to the counting is the bottleneck of their method, that can be overcome by our algorithms.

We are currently working on two main improvements. Although it only needs to be performed once, the generation of the list of all k-motifs is still demanding. We are currently working on the improvement of this step. In addition, for large motifs it could be desirable to avoid the scan of the entire network whereas the motif is not present in the network. This could be done, for example, in a forward way, first counting the motifs of a given size k, and then pruning the list of larger motifs with size $k' > k$, based on the presence or absence of their sub-motifs of size k.

Acknowledgement

This work has been supported by the French Agence Nationale de la Recherche under grant NeMo ANR-08-BLAN-0304-01. The authors want to thank all the members of the NeMo project for their advice and helpful discussions.

References

[1] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, Networks motifs in the transcriptional regulation network of *escherichia coli*. *Nat. Genetics*, 31:64–68, 2002.

[2] S. Wuchty, Z. N. Oltvai, and A. L. Barabasi, Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nat. Genet.*, 35:176–179, Oct 2003.

[3] Y. Chen and N. V. Dokholyan, The coordinated evolution of yeast proteins is constrained by functional modularity. *Trends Genet.*, 22:416–419, Aug 2006.

[4] B. Papp, C. Pál, and L. D. Hurst, Dosage sensitivity and the evolution of gene families in yeast. *Nature*, 424:194–197, Jul 2003.

[5] N. Bhardwaj and H. Lu, Co-expression among constituents of a motif in the protein-protein interaction network. *J Bioinform Comput Biol*, 7:1–17, Feb 2009.

[6] V. Spirin and L. A. Mirny, Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA*, 100(21):12123–8, 2003.

[7] V. Lacroix, C.G. Fernandes, and M. F. Sagot, Motif search in graphs: application to metabolic networks. *IEEE/ACM Trans Comput Biol Bioinform*, 3:360–8, 2006.

[8] V. Lacroix, C.G. Fernandes, and M.-F. Sagot, Reaction motifs in metabolic networks. In *Proceedings of WABI 2005*, volume 3692 of *Algorithm in Bioinformatics*. Lecture notes in Computer Science, 2005.

[9] S. Schbath, V. Lacroix, and M. F. Sagot, Assessing the exceptionality of coloured motifs in networks. *EURASIP J Bioinform Syst Biol*, page 616234, 2009.

[10] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, Networks motifs: simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[11] S. Mangan and U. Alon, Structure and function of the feed-forward loop network motif. *PNAS*, 100:11980–11985, 2003.

[12] E. Birmelé, Detection of network motifs by local concentration. In *JOBIM*, 2009. www.citebase.org/abstract?id=oai:arXiv.org:0904.0365.

[13] F. Picard, J.-J. Daudin, M. Koskas, S. Schbath, and S. Robin, Assessing the exceptionality of network motifs,. *J. Comp. Biol.*, 15(1):1–20, 2008.

[14] S. Wernicke and F. Rasche, FANMOD: a tool for fast network motif detection. *Bioinformatics*, 22:1152–1153, May 2006.

[15] Z. R. Kashani, H. Ahrabian, E. Elahi, A. Nowzari-Dalini, E. S. Ansari, S. Asadi, S. Mohammadi, F. Schreiber, and A. Masoudi-Nejad, Kavosh: a new algorithm for finding network motifs. *BMC Bioinformatics*, 10:318, 2009.

[16] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20:1746–1758, Jul 2004.

[17] W Kocay, An extension of Kellys lemma to spanning subgraphs. *Congr. Num.*, 31:109–20, 1981.

[18] J. A. Grochow and M. Kellis, Network motif discovery using subgraph enumeration and symmetry-breaking. In *RECOMB*, pages 92–106, 2007.

[19] P. Erdös, Some remarks on the theory of graphs. *Bull. Am. Math. Soc.*, 53:292–94, 1947.

[20] J. Park and M. Newman, The origin of degree correlations in the internet and other networks. *Phys. Rev. E*, 68:026112, 2003.

[21] F. Chung and L. Lu, The average distances in random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, 99:15879–15882, 2002.

[22] J.-J. Daudin, F. Picard, and S. Robin, A mixture model for random graphs. *Stat. Comput.*, 18(2):173–83, Jun 2008.

[23] K. Nowicki and T.A.B. Snijders, Estimation and prediction for stochastic block-structures. *J. Amer. Statist. Assoc.*, 96:1077–87, 2001.

[24] G. Nuel, Cumulative distribution function of a geometric Poisson distribution. *J. Statis. Comput. Simul.*, 78(3):385–94, 2008. DOI: 10.1080/10629360600997371.

[25] N. Przulj, Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23:e177–183, Jan 2007.

[26] D. Zhu and Z. S. Qin, Structural comparison of metabolic networks in selected single cell organisms. *BMC Bioinformatics*, 6:8, 2005.

[27] Z. Wang, X. G. Zhu, Y. Chen, Y. Li, J. Hou, Y. Li, and L. Liu, Exploring photosynthesis evolution by comparative analysis of metabolic networks between chloroplasts and photosynthetic bacteria. *BMC Genomics*, 7:100, 2006.

[28] M. Parter, N. Kashtan, and U. Alon, Environmental variability and modularity of bacterial metabolic networks. *BMC Evol. Biol.*, 7:169, 2007.

[29] O. Ratmann, O. Jrgensen, T. Hinkley, M. Stumpf, S. Richardson, and C. Wiuf, Using likelihood-free inference to compare evolutionary dynamics of the protein networks of *h. pylori* and *p. falciparum*. *PLoS Comput. Biol.*, 3(11), 2007.

[30] M. Koyüturk, Y. Kim, S. Subramaniam, W. Szpankowski, A. Grama, Detecting Conserved Interaction Patterns in Biological Networks. *J. Comp. Biol.* 13(7) 1299-1322, 2006.