

RANGI: A Fast List-Colored Graph Motif Finding Algorithm

Ali Gholami Rudi, Saeed Sahrivari, Saeed Jalili, Zahra Razaghi Moghadam Kashani

Given a multiset of colors as the query and a list-colored graph, i.e. an undirected graph with a set of colors assigned to each of its vertices, in the NP-hard list-colored graph motif problem the goal is to find the largest connected subgraph such that one can select a color from the set of colors assigned to each of its vertices to obtain a subset of the query. This problem was introduced to find functional motifs in biological networks. We present a branch-and-bound algorithm named RANGI for finding and enumerating list-colored graph motifs. As our experimental results show, RANGI's pruning methods and heuristics make it quite fast in practice compared to the algorithms presented in the literature. We also present a parallel version of RANGI that achieves acceptable scalability.

Index Terms—Branch-and-bound algorithms, Parallel algorithms, List-colored graphs, Protein interaction networks.

1. Introduction

With the availability of complex biological networks many researchers began analyzing them to extract meaningful patterns, such as "network motifs". Although the original definition of network motifs was concerned with the topology of the motifs and their interconnections [1], recently Lacroix et al. introduced functional, or topology-free network motifs based on the functionalities of nodes [2]. In this new definition of network motifs, a set of functionalities is assigned to each node of the network; for instance in metabolic networks, where nodes represent reactions with edges between reactions that occur successively, these functionalities could represent the types of these reactions. The problem of finding functional motifs in biological networks can be modeled as finding a subgraph in list-colored graphs, in which a set of colors is assigned to each vertex. Given a multiset of colors as the query, the goal is to find the largest connected subgraph of the input list-colored graph such that it is possible to select a color from the set of colors assigned to each vertex in the subgraph to obtain a subset of the query.

*A.G. Rudi, S. Sahrivari, S. Jalili, and Z.R.M. Kashani are with the Tarbiat Modares University, 14115-194 Tehran, Iran.
E-mail: {gholamirudi, s.sahrivari, sjalili}@modares.ac.ir, razaghi@ibb.ut.ac.ir.*

In this paper we present a branch-and-bound algorithm named RANGI for finding list-colored graph motifs. RANGI enumerates subgraphs of the input graph based on a subgraph enumeration algorithm, like Wernicke and Rasche’s FANMOD [3] and Razaghi et al.’s Kavosh [4], both of which have been used for finding topological motifs. RANGI augments these enumeration algorithms with pruning methods and heuristics to reduce the size of the search space. We probabilistically show that these methods are quite effective in reducing the size of the search space. RANGI enumerates all motifs of the maximum size, which allows defining different criteria for selecting only one of these motifs as the most significant. We also present a parallel version of our algorithm to take advantage of multi-core processors.

We evaluate our algorithm on three protein interaction networks for yeast, fly and human assembled by Bruckner et al. [5], which has been widely used for evaluating colored and list-colored graph motifs [5–8]. In these experiments we try to find protein complexes in a species based on the complexes of another. Our experimental results show that RANGI’s pruning methods are quite effective and it is significantly faster than Betzler et al.’s color-coding-based dynamic programming algorithm for finding list-colored graph motifs [7]. Also the parallel version of our algorithm achieves an average speedup of more than 2.5 on a quad-core processor, proving RANGI acceptably scalable.

This paper is organized as follows: In Section 2, we review some of the related studies in the literature. Next, we define the notations used in this paper and formally describe the list-colored graph motif problem in Section 3. In Section 4, we present RANGI, describe its pruning methods and heuristics, and analyze it. In Section 5, we present a parallel version of RANGI. Then in Section 6, we report our experimental results and finally in Section 7, we conclude this paper.

2. Related Work

Lacroix et al. introduced functional motifs for metabolic networks and proved that the problem of finding them in list-colored graphs is NP-complete [2]. Some researchers investigated a special case of this problem on vertex-colored graphs, where each vertex has a single color. Fellows et al. proved the NP-completeness of this problem for restricted classes of graphs and presented an FPT algorithm for it [9]. Guillemot and Sikora presented an algorithm by reducing the graph motif to the Multilinear Detection problem to obtain an $O(n^{O(1)}4^k)$ algorithm to solve the graph motif problem [8,10]. Koutis used an extended version of this technique, *Constrained* Multilinear Detection, to present an algorithm with complexity $O(n^{O(1)}2.54^k)$ [11], and later Björklund et al. improved it to $O(n^{O(1)}2^k)$ [12]. Dondi et al. discussed three variants of this problem and the fixed-parameter tractability of them [13]. Ambalath et al. proved the NP-completeness of this problem on simple classes of graphs and studied the possibility of decomposing restricted graphs for solving the graph motif problem on smaller instances [14].

However, few researchers have studied the problem of list-colored graph motifs after its introduction by Lacroix et al. [2]. Bruckner et al. proposed an algorithm called TORQUE based on integer linear programming and color-coding [15] to solve an extension of the list-colored

graph motif problem, where the motif can contain uncolored vertices [5]. Blin et al. presented the GraMoFoNe algorithm that reduces this extension of the list-colored graph motif to linear pseudo-boolean optimization problem and experimentally showed that GraMoFoNe outperforms TORQUE [6].

Betzler et al. presented algorithms based on color-coding and dynamic programming for finding list-colored graph motifs with the highest maximum tree weight [7]. For parameter $|M|$ (the query size) they presented an $O(10.88^{|M|} \cdot m)$ algorithm for graphs with m edges, and for parameter $|S|$ (the motif size) they presented an algorithm with time complexity $O(29.6^{|S|} \cdot |S| \cdot m)$.

In this paper, we present the branch-and-bound RANGI algorithm for solving the list-colored graph motif problem. Although the running time of RANGI depends on the number of connected induced subgraphs of the input graph, RANGI's heuristics can prune large sections of the search space, as we shall show in Section 4.4. Our experimental results of Section 6 also show that RANGI is considerably faster than Betzler et al.'s algorithm and GraMoFoNe in practice. Also unlike Betzler et al.'s algorithm, RANGI enumerates all motifs of the optimal size.

3. Preliminaries

We assume a biological network is represented as an undirected graph G , with vertex set $V(G)$ and edge set $E(G)$. For brevity, we use n instead of $|V(G)|$ to denote the number of vertices in a graph, when there is no confusion. The graph H is a subgraph of G , if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An induced subgraph of G on the set of vertices H , $G[H]$, is a subgraph of G with vertex set H that contains all edges between the vertices of H in G . A list-colored graph is an undirected graph with a set of colors, $\text{col}(v)$, associated with each of its vertices, v .

List-Colored Graph Motif Problem

Given a list-colored undirected graph G and a multiset of colors M as the query, the list-colored graph motif problem is defined as finding the sets $S \subseteq V(G)$ and $M' \subseteq M$ with maximum cardinality such that the induced subgraph $G[S]$ is connected and there is a bijective mapping $f: S \rightarrow M'$ so that for every $v \in S$, $f(v) \in \text{col}(v)$ [2].

Betzler et al. have shown that the problem of finding list-colored graph motifs when M is a multiset can be converted to the problem of finding motifs in a new graph where the query is a set [7]. Therefore, for simplicity we restrict ourselves to the case where M is a set, as they did. However, only minor modifications are needed to handle multiset queries in RANGI.

4. The RANGI Algorithm

As depicted in Figure 4.1, for finding list-colored graph motifs RANGI enumerates possible motifs of the input graph and checks each of them to see if it is possible to select a color from the set of colors assigned to each of its vertices to obtain a subset of the query. The search space, i.e. all connected subgraphs of the input graph, can be very large. In order to reduce the

time required for searching the whole search space, RANGI prunes large sections of the search space to discard subgraphs that cannot be a motif.

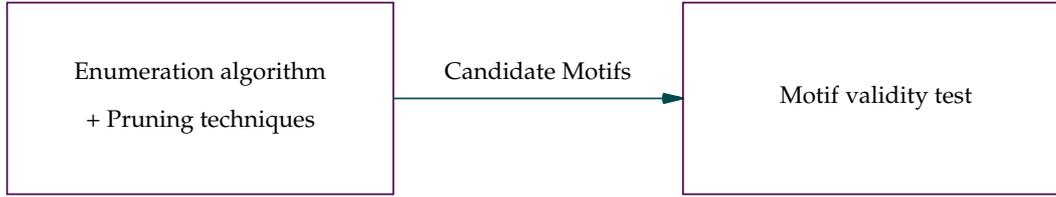


Figure 4.1 - The overall flow of RANGI

In the rest of this section, we first explain how RANGI enumerates the connected subgraphs of a graph and how it uses pruning techniques to prune parts of the search space. Then, we explain four of the pruning techniques and heuristics used in RANGI. In Section 4.3 we describe how RANGI checks the validity of the candidate motifs. Finally, although the utility of these techniques are easier to verify experimentally, we use a simple probabilistic analysis in Section 4.4 to show how effective two of these techniques can be.

4.1. Subgraph Enumeration

The way RANGI enumerates candidate motifs depends on its enumeration algorithm. RANGI can use any subgraph enumeration algorithm as long as it has the following two properties (for finding subgraphs of size k):

1. *Arbitrary Initial Vertex (AIV)*: Given any starting vertex, the enumeration algorithm should enumerate all subgraphs of size k that contain this vertex.
2. *Recursive Vertex Inheritance (RVI)*: Subgraphs should be enumerated recursively; each recursive call receives a set of vertices and should recursively enumerate only the subgraphs that contain these vertices.

To enumerate all subgraphs of the input graph G , we can define an ordering $R = (r_1, r_2, \dots, r_n)$ on the vertices of G , where $|V(G)| = n$. Subgraphs are enumerated in n steps: in step i , all subgraphs of size k are enumerated in the induced subgraph of G on vertices r_i, r_{i+1}, \dots, r_n starting from the vertex r_i . Property *AIV* ensures that all subgraphs of size k are reported exactly once; the subgraphs that contain r_i and no vertex r_j for $j < i$ are only reported in step i .

RANGI uses a similar procedure for enumerating all subgraphs. However, it modifies this algorithm to implement its pruning techniques for finding list-colored graph motifs: 1) The set R may contain only some of the vertices of G . Property *AIV* of the enumeration algorithm makes sure only the subgraphs containing any of the vertices in R are reported. 2) If the recursive part of the enumeration algorithm is called with a set of vertices that no motif can contain, the recursive call is terminated early. Property *RVI* makes sure only the subgraphs that contain the given set of vertices are discarded. Note that since the size of the optimal motif is not known in advance, RANGI should try different motif sizes until it finds the optimal size: starting from motifs of size 2, it gradually tries larger motifs until it finds a number k such that

there exist motifs of size k but no motif of size $k + 1$.

FANMOD [3] and Kavosh [4] find topological motifs by enumerating connected subgraphs of the given size just as RANGI does. Their enumeration algorithms have the *AIV* and *RVI* properties. We shall experimentally evaluate their enumeration algorithm in Section 6. Kavosh enumerates the subgraphs of a graph based on its Breadth-First Search tree. For the set of vertices at distance i from the initial vertex, it tries every combination of the neighbors of the vertices at distance $i - 1$. FANMOD, on the other hand, maintains a set of the neighbors of the vertices previously selected for the motif and tries adding each of them to the selected vertices to recursively find all containing motifs.

4.2. Branch-and-bound Heuristics

RANGI uses the following pruning methods and heuristics.

Pruning Based on the Selected Colors

Based on the *RVI* property of the enumeration algorithm (as explained in Section 4.1), RANGI skips the recursive calls of the enumeration algorithm that receive a set of vertices that cannot appear in a motif. It uses the set of colors assigned to the vertices to identify such sets of vertices.

By definition, for the list-colored graph motif S , there is a function f that to each vertex of S assigns one of its colors so that $f(S) = \{f(v) \mid v \in S\}$ is a subset of the query. A similar relation holds for every subset of S too: for $T \subseteq S$, $f(T)$ should also be a subset of the query of size $|T|$ using the same mapping function f as S . Since the query is colorful, the set $f(T)$ should contain $|T|$ distinct colors. Defining $C(T)$ as:

$$C(T) = \bigcup_{v \in T} \text{col}(v)$$

$|C(T)|$ should be no smaller than $|T|$, since $f(T) \subseteq C(T)$.

Based on this argument, if $|C(T)| < |T|$, it cannot be part of a motif. RANGI skips the recursive calls of the enumeration algorithm with such sets of vertices.

This heuristic is actually a simplified and more computationally efficient version of the motif validity test of Section 4.3; instead of finding the mapping function f , we only count the number of distinct colors in $C(T)$. The overhead of performing the motif validity test in each recursive call of the enumeration algorithm makes RANGI substantially slower, despite its higher accuracy for detecting invalid subgraphs. To solve this dilemma, we define a branching threshold: only if a recursive call of the enumeration algorithm produces more than this many branches (the number of times it calls the recursive function directly), we perform the motif validity test and otherwise we perform the simplified test. If the recursive call creates many branches, the overhead of the motif validity test should be tolerable.

Limiting Initial Vertices

Let M be the query and S be any motif of size k in the graph G . Since there should exist a one-to-one correspondence between the vertices of S and the colors of a subset of M , the vertices in the motif should contain at least k different colors (note that here M is a set; when M is a multiset, a similar but weaker condition holds). So given any set of colors C of size $|M| - k + 1$ where $C \subseteq M$, each motif contains a vertex v so that $\text{col}(v) \cap C \neq \emptyset$, otherwise at most $k - 1$ colors can appear in S . Hence, instead of searching all vertices in a graph, we can search those that have common colors with the set C . By the *AIV* property of the subgraph enumeration algorithm, we can start the enumeration from these sets of vertices to obtain all subgraphs that contain at least one of them. More precisely, we define R as follows:

$$R = \{v \mid v \in V(G), \text{col}(v) \cap C \neq \emptyset\}$$

Where $C \subseteq M$ and $|C| = |M| - k + 1$. Then in RANGI we use R as the list of initial vertices.

This heuristic works for an arbitrary choice of $C \subseteq M$ with the mentioned size. However, we can improve it by applying a heuristic for finding the set C : Among all colors in M , we choose the ones that have the lowest frequency in the vertices of the graph to further decrease the number of vertices to be chosen as the initial vertices in RANGI. This heuristic is quite effective, especially when the motif size k is close to the query size $|M|$, since the size of C depends on their difference. Besides, since the time needed for finding motifs grows almost exponentially with k , the improvement becomes more noticeable for larger values of k .

Furthermore, a vertex can appear in a motif of size k only if there are at least k distinct colors in the set of vertices at distance at most $k - 1$ from it. We remove initial vertices that fail this condition. Note that this heuristic implicitly discards components of the input graph with less than k different colors in the sets of colors assigned to its vertices.

Ordering Initial Vertices

Although RANGI works for any ordering of the initial vertices for enumeration $R = (r_1, r_2, \dots)$, the order of vertices in R can change the running time of RANGI significantly. We define G_i as the induced subgraph on vertices $V(G) \setminus \{r_1, r_2, \dots, r_{i-1}\}$. When enumerating from the i -th initial vertex, we are actually searching for subgraphs of size k starting from r_i in G_i . The order of the vertices in R is important for two reasons: First, if G_i graphs for $1 \leq i \leq |R|$ are almost of the same complexity (we simply define complexity as the time it takes for the enumeration algorithm to search it), all threads in the parallel version of our algorithm can explore the graph, without becoming idle waiting for other threads. Furthermore, in a good ordering (which enumerates subgraphs with less complexity), time consuming branches can be trimmed early.

We use a simple heuristic for ordering the vertices in R : we sort the vertices based on their degrees so that the vertices with the lowest degrees are selected as graph roots earlier. Although the main reason for introducing this heuristic was better threaded speedup, surprisingly it worked quite well in reducing the running time of the single-threaded

algorithm. For the threaded case, this ordering would prevent choosing vertices with large degrees early which may result in dense graphs for the first few G_i graphs.

Early Motif Detection

As mentioned in Section 4.1, RANGI tries different motif sizes until it finds the motifs of the optimal size. Since finding larger motifs usually takes longer (based on experimental evaluations), RANGI starts from motifs of size two and gradually increases the size of the motifs until it finds a number k so that there is no motif of size $k + 1$. However fully exploring motifs of size k is unnecessary when k is not the optimal size. Also to make sure k is optimal, we need to ensure that there are no motifs of size $k + 1$ by searching for them. RANGI improves these steps based on the fact that a motif of size $k + 1$ can be created by adding a vertex to a motif of size k .

When RANGI finds a motif of size k , it adds each of the neighbors of the set of vertices in the motif to see if a motif of size $k + 1$ can be created. When such a motif is found, RANGI stops searching for motifs of size k and starts searching for motifs of size $k + 1$. On the other hand, any motif of size $k + 1$ can be created by adding a vertex to a motif of size k . So if after searching for motifs of size k , RANGI could not find any motif of size $k + 1$ by adding a vertex to any of the motifs of size k , it assumes motifs of size k are the optimal motifs.

4.3. Motif Validity Test

For each candidate subgraph S , a bipartite graph like Figure 4.2 is created: One part consists of vertices in S , and the other has a vertex for each color in the query. Each vertex like v is adjacent to the colors in its color list, $\text{col}(v)$. If this graph has a matching of size $|S|$, it is a valid motif and there exists a mapping between vertices of S and a subset of M , as required by the definition of list-colored graph motifs. RANGI uses Hopcroft–Karp bipartite maximum matching algorithm in this step with time complexity $O(|M|^{5/2})$ [16].

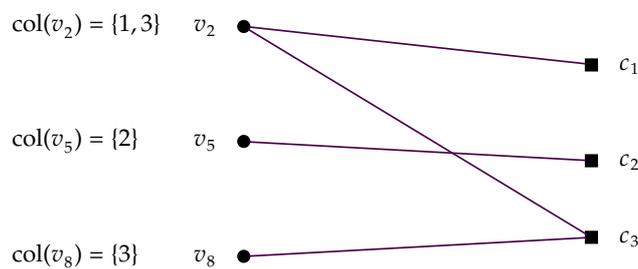


Figure 4.2 – The bipartite graph for testing the validity of a motif of size 3

4.4. Complexity Analysis

The actual running time of RANGI for finding motifs of size k in a graph depends on the number of its connected induced subgraphs of size at most k . In complete graphs there are $O(n^k)$ such subgraphs, which grows very fast in respect to k . However the number of connected induced subgraphs of sparse graphs can be much smaller; for degree bounded graphs with

maximum degree , one can show that there are $n4^k(-1)^k$ connected subgraphs of size at most k [17]. But these upper bounds merely describe the size of the search space and do not show how RANGI's pruning techniques can reduce it.

In the rest of this section we probabilistically analyze RANGI's pruning techniques and how much they reduce the size of the search space. The performance of RANGI's pruning techniques depends on the input graph and the distribution of colors among its vertices; we assume the input graph is vertex-colored (the size of the set of colors assigned to each vertex is one) and the colors are uniformly distributed among the vertices of the graph. Also we assume searching for motifs of size exactly k , the size of the query. Theorem 4.1 probabilistically shows how RANGI's first heuristic reduces the size of the search space.

Theorem 4.1: For finding colored motifs of size k in a uniformly colored graph, RANGI probabilistically checks at most $e^{1-k} \cdot k^{1/2}$ of the connected induced subgraphs of size k .

Proof: Let A be the set of all connected induced subgraphs of size k in the colored graph G . Without any heuristics, all subgraphs in A are explored by the enumeration algorithm. RANGI's first heuristic discards any subgraph with duplicate vertices, so the probability of not being discarded $P(H)$ for any subgraph $H \in A$ is (note that $|H| = k$):

$$P(H) = \frac{k!}{k^k}$$

That is out of the k^k ways of colorings of H , only the ones in which all of the colors appear are preserved ($k!$ which is the number of permutations of k distinct colors). As a result, the expected number of subgraphs checked by RANGI N equals:

$$N \leq \frac{k!}{k^k} |A|$$

By the Stirling's approximation of factorials $k! \leq k^{k+1/2}e^{1-k}$, we have:

$$N \leq \frac{e\sqrt{k}}{e^k} |A|$$

On the other hand, only $1/k$ of the vertices of the input graph are selected as the initial vertex by the second heuristic in a uniformly colored graph. Depending on the input graph, this may reduce the size of the search space to a factor of $1/k$, without the first heuristic. For a complete graph, for instance, the number of subgraphs checked by RANGI without the first heuristic is ($m = n/k$):

$$N \leq \frac{\binom{n}{2} - \binom{n-m}{2}}{\binom{n}{2}} |A|$$

Which implies RANGI enumerates at most $2/k|A|$ subgraphs. The cumulative effect of the first two heuristics depends on the input graph; The first heuristic (Theorem 4.1) reduces the number of motif validity tests by pruning internal branches of the search tree (the tree formed by the recursive calls of the enumeration algorithm of RANGI), while the second decreases the

number of initial vertices and hence prunes the top-level branches of the search tree.

5. The Parallel Version of RANGI

In this section we present a parallel version of RANGI to make effective use of multi-core processors. In RANGI there is no dependency between subgraph enumerations with different initial vertices, except that previously selected initial vertices should be deleted from the input graph and should not appear in any later subgraphs. We use this property in Section 5.1 to implement the ThreadedRANGI algorithm. Then in Section 5.2, we try to improve it by using a more fine-grained division of the enumeration process between threads.

5.1. The Threaded RANGI

Let $R = (r_1, r_2, \dots)$ be the list of initial enumeration vertices in RANGI as discussed in Section 4.1. As explained in Section 4.1, in the i -th iteration of RANGI, the enumeration algorithm is called from the initial vertex r_i in the graph $G_i = G \setminus \{r_1, \dots, r_{i-1}\}$. The lack of any dependency between the exploration of G_i from r_i makes it possible to perform the enumeration in parallel by dividing the enumeration of these subgraphs in separate execution threads.

Additionally, in the list-colored graph motif problem, we try to find motifs of the largest possible size. Since we do not know the size of the optimal motif in advance, RANGI tries different motif sizes as explained in Section 4.1. In RANGI, not only we can distribute the initial vertices among threads, but we can do so for finding motifs of different sizes.

The optimal division of initial vertices between threads is difficult to determine statically though. So we maintain a lock-protected list of free initial vertices from which different threads query the availability of a vertex when they become idle as shown in Figure 5.1.

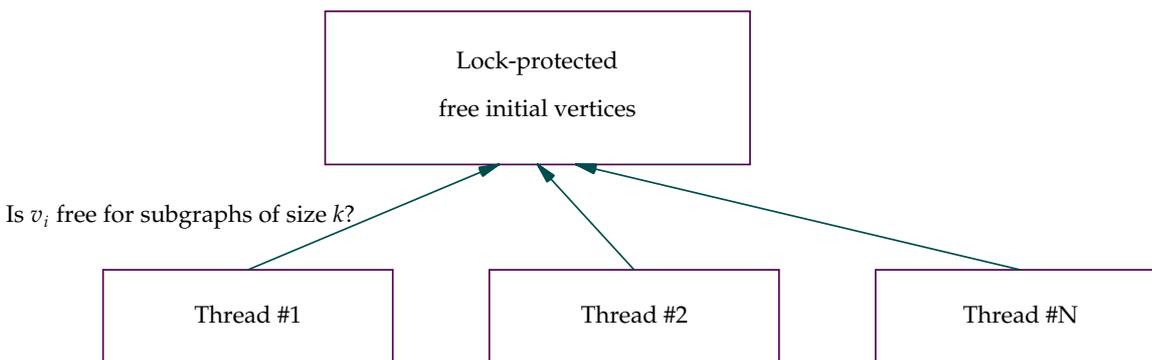


Figure 5.1 - Dynamic division of initial vertices among threads in Threaded RANGI

Algorithm 5.1 demonstrates how each thread processes initial vertices. It takes the input graph G , the ordered lists of initial vertices for different motif sizes R_s , and the largest motif size to find $maxk$. **Enumerate** (H, v, k) starts enumerating subgraphs of size k from vertex v in H .

Algorithm 5.1 - **ThreadedRANGI**($G, Rs, maxk$)

1. for k from 2 to $maxk$
2. let H be a copy of G
3. for r_i in $Rs[k]$
4. if r_i for subgraphs of size k is free
5. **Enumerate**(H, r_i, k)
6. remove r_i from H

We implemented another optimization to improve **ThreadedRANGI**(): when one of the threads finds a motif of size k , the motifs of the maximum size cannot be any smaller than k . Based on this observation, as soon as one of the threads finds a motif of size k , all threads searching for smaller motifs abandon their search and request initial vertices for subgraphs of size k or larger. This optimization can result in speedups larger than the number of processors, compared to the single threaded version of RANGI.

5.2. The XAIV Extension

Judging by our experimental results, our heuristic to limit the number of initial vertices is so effective that for some graphs very few initial vertices are considered, especially when the motif size is close to the query size, even though we search different subgraph sizes in parallel. To improve processor utilization for such graphs, we changed the FANMOD enumeration algorithm to support an extended version of the *AIV* property of Section 4.1, i.e. to search for subgraphs from a pair of adjacent vertices. We refer to this extended property as *XAIV* in the rest of this paper. In ThreadRANGI/XAIV (ThreadedRANGI augmented with the *XAIV* property) instead of distributing the initial vertices, we distribute the (r_i, n_j) vertex pairs among threads, where r_i is the i -th initial vertex and n_j is its j -th neighbor. When searching for motifs starting from the pair (r_i, n_j) , we start the enumeration algorithm from the pair of vertices r_i and n_j in the graph $G_{ij} = G_i \setminus \{n_1, n_2, \dots, n_{j-1}\}$ where G_i is defined as in 5.1. In Section 6.2 we evaluate this extension experimentally.

6. Experimental Results

As most of the previous colored and list-colored graph motif experiments in the literature including Betzler et al. [5–8], we used three protein interaction networks assembled by Bruckner et al. for yeast (5,430 proteins, 39,936 interactions), fly (6,650 proteins, 39,936 interactions), and human (7,915 proteins, 21,275 interactions) [5]. We also acquired their query complexes for yeast, fly, human, bovine, mouse, and rat.¹ For each query, we gave a color to each of its proteins. Then we assigned the colors of the query proteins to the proteins in the network whose BLAST E-values were lower than 10^{-7} . Table 6.1 summarizes the size of our protein complexes in any of the 6 species and Table 6.2 summarizes the size of the optimal motifs for

¹ Available at <http://igm.univ-mlv.fr/AlgoB/gramofone/>

those queries in the available networks.

Table 6.1 – The size of query complexes from different species

Query Set	$2 \leq M < 4$	$4 \leq M < 8$	$8 \leq M < 16$	$16 \leq M < 32$	$32 \leq M < 64$	$64 \leq M $
human	522	272	110	27	6	5
fly	39	37	29	14	4	1
yeast	99	98	57	18	13	6
rat	95	49	5	1	0	0
bovine	8	6	4	0	0	1
mouse	136	44	15	0	2	0

Table 6.2 – The size of the optimal motifs in different networks

Network	$2 \leq S < 4$	$4 \leq S < 8$	$8 \leq S < 16$	$16 \leq S $
human	361	157	25	12
fly	435	32	6	3
yeast	407	140	40	9

We implemented RANGI in POSIX C with Kavosh and FANMOD (ESU) enumeration algorithms.² The tests were performed on a desktop machine with an Intel Core i7 2600 processor and 16GB of RAM. The C++ implementation of Betzler et al.'s algorithm (GRAM) were obtained from the address included in their paper [7]. For GraMoFoNe, we used their Cytoscape plugin [6].

6.1. Comparison With Existing Software

We compare 5 different configurations of RANGI and GRAM in Tables 6.3 and 6.4. In these tables "RANGI/FANMOD" and "RANGI/Kavosh" indicate the RANGI algorithm with FANMOD and Kavosh enumeration algorithms respectively, "Exhaustive" is just like RANGI/FANMOD with all RANGI pruning methods and heuristics disabled and "GRAM/|S|" and "GRAM/|M|" indicate Betzler et al.'s algorithm with parameters motif size and query size respectively. For each test, the programs were given 10 minutes to find the optimal motif; the tests taking longer or running out of memory were assumed unsolved. The results of these tables are grouped based on the motif size $|S|$, except for the tests with queries of size larger than 64; these tests were separated because the GRAM program is limited to queries of size at most 64.

² The source code is available at <http://litcave.rudi.ir/rangi.html>

Table 6.3 – Comparison of the percentage of solved instances

Algorithm	$4 \leq S < 8$	$8 \leq S < 16$	$16 \leq S $	$64 < M $
RANGI/FANMOD	100.0 (328/328)	100.0 (68/68)	91.7 (11/12)	75.0 (18/24)
RANGI/Kavosh	100.0 (328/328)	100.0 (68/68)	83.3 (10/12)	70.0 (17/24)
Exhaustive	98.8 (324/328)	86.8 (59/68)	66.7 (8/12)	54.2 (13/24)
GRAM/ M	99.1 (325/328)	75.0 (51/68)	50.0 (6/12)	N/A
GRAM/ S	99.7 (327/328)	67.7 (46/68)	33.3 (4/12)	N/A

Table 6.3 lists the percentage of solved instances. RANGI with FANMOD enumeration solves all but one of the instances with queries of size less than 64. Among instances with queries of size at least 65, only 6 were not completed in 10 minutes using RANGI/FANMOD. Table 6.4 lists the average running time of different algorithms for the instances solved by at least one of the algorithms. Note that we assumed a running time of 10 minutes (the value of the timeout) for unsolved instances, so the values, specially for the last two columns, do not represent the relative speed of the algorithms; only that slower algorithms get closer to 10 minutes. Nevertheless, the difference between the Exhaustive algorithm and RANGI seem to follow our theoretical results of Section 4.4.

Table 6.4 – The average running time of the algorithms (seconds)

Algorithm	$4 \leq S < 8$	$8 \leq S < 16$	$16 \leq S $	$64 < M $
RANGI/FANMOD	0.00	0.06	14.91	15.77
RANGI/Kavosh	0.01	3.90	55.48	69.19
Exhaustive	7.95	82.60	209.455	166.64
GRAM/ M	8.21	165.477	277.17	N/A
GRAM/ S	5.83	224.95	386.41	N/A

Figure 6.1 shows the number of unsolved instances for different values of time limit for RANGI/FANMOD and GRAM/|M| for instances with queries of size at most 64. RANGI performs much better than Betzler et al.'s algorithm; it solves almost all of the instances within ten seconds. Among RANGI/FANMOD's 7 unsolved instances of Table 6.3, ThreadedRANGI solves one of them within the 10-minute time limit. The optimal motifs of the remaining 6 instances were very large: the largest motif found by ThreadedRANGI for these instances were of size 41 to 83.

We also compared RANGI with Blin et al.'s GraMoFoNe [6] for finding motifs in yeast network using 13 queries from human network that took more than 0.01 seconds in RANGI (8 of these queries had more than 26 proteins). The results are shown in Table 6.5. For testing we allowed no insertions, but a sufficiently large number of deletions to include the optimal motif. However, GraMoFoNe did not report the optimal motif for some of the instances. But nevertheless, RANGI seems much faster than GraMoFoNe.

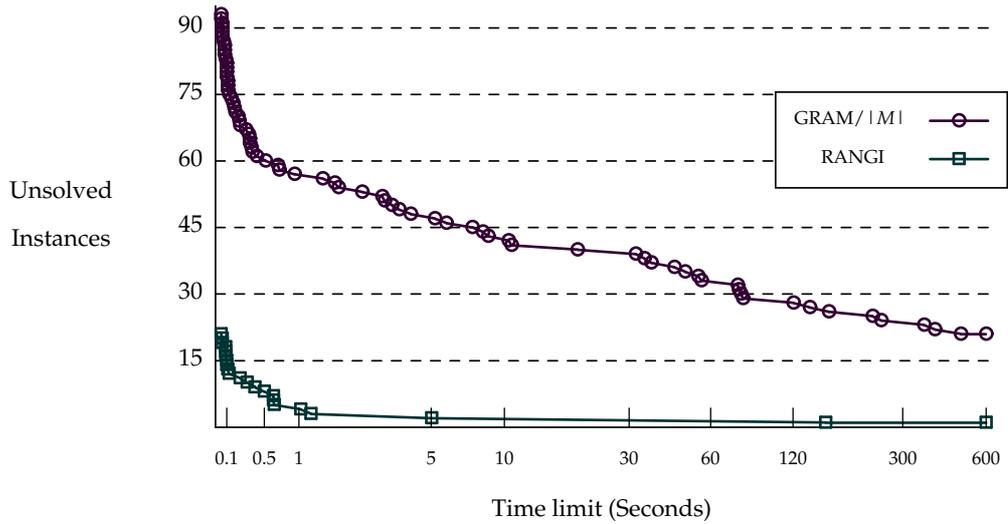


Figure 6.1 – Unsolved instances for different time limit values for instances in which $|M| \leq 64$

Table 6.5 – The comparison of running time between RANGI and GraMoFoNe

Program	Solved	Average (Seconds)
RANGI	76.9% (10/13)	150.7
GraMoFoNe	30.8% (4/13)	431.9

6.2. Evaluating the Parallel Version of RANGI

As discussed in Section 5, we modified the FANMOD enumeration algorithm to implement the XAIV property. In Table 6.2, we compare the average speedup of ThreadedRANGI and ThreadedRANGI/XAIV on tests that took more than half a second in RANGI. The more fine-grained division of the enumeration among threads in ThreadedRANGI/XAIV has made it more scalable as expected.

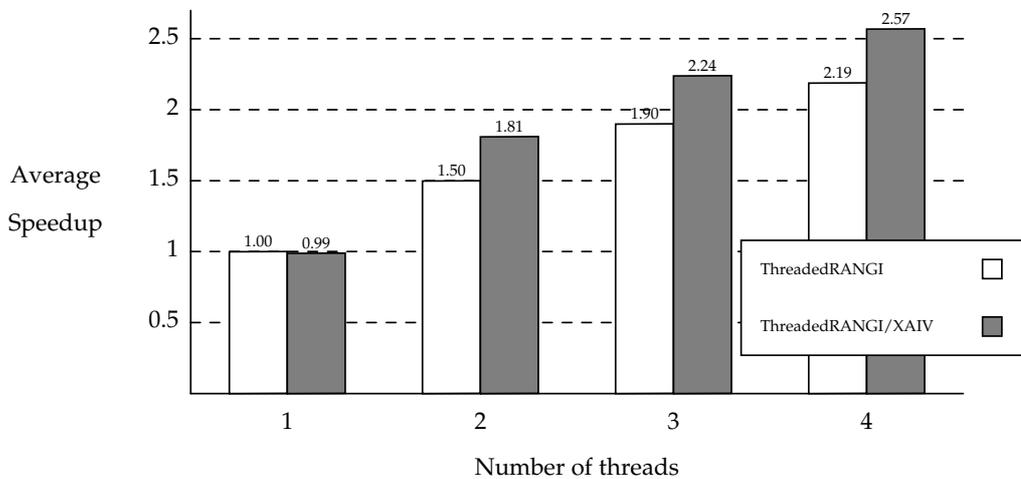


Figure 6.2 – Average speedup of ThreadedRANGI and ThreadedRANGI/XAIV

Figure 6.3 shows the average speedup of ThreadedRANGI/XAIV for tests that took more than half of a second in RANGI. Interestingly, the maximum amount of speedup compared to the single threaded algorithm was more than the number of cores for some of the tests. Even one of the unsolved tests in Section 6.1 was solved in less than 2 minutes (not included in this figure). As discussed in Section 5, the reason for this observation is that threads searching for small motifs abandon their search when a thread finds a larger motif.

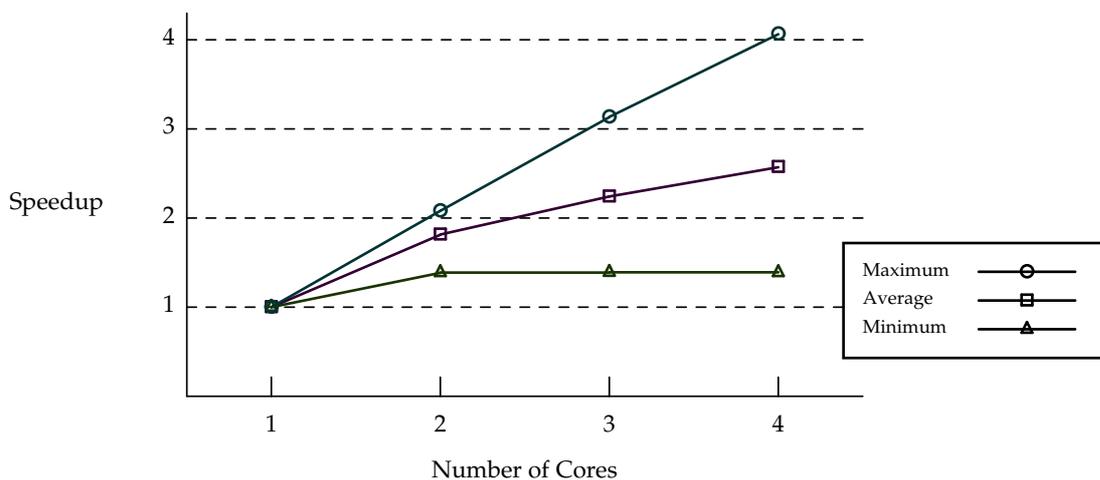


Figure 6.3 - Multi-core speedup of ThreadedRANGI/XAIV

6.3. Functional Enrichment of the Motifs

Unlike Betzler et al.'s GRAM and Bruckner et al.'s TORQUE [5,7], RANGI finds all motifs of the maximum size in a graph. If only one of these possibly many motifs should be selected as the most significant motif, different selection criteria could be defined. GRAM tries to find the motif with the greatest maximum weight spanning tree. We define three other weighting methods: the sum of the weights of the edges, the minimum weight of the edges, and the number of the edges of the motif, and evaluate the biological significance of the selected motifs based on any of these weighting methods.

We use two measures for evaluating the significance of the motifs: *complex similarity* and *functional enrichment evaluation*. For the complex similarity measure, we define the similarity between a motif and a complex as the ratio of the number of their common proteins to the total number of proteins in the complex. Then for each motif we define complex similarity as its maximum similarity to any of the complexes of the network. We use the same protein complexes as used in the previous parts of this section for yeast, human and fly networks.

For evaluation of the functional enrichment of the motifs, we followed Betzler et al. [7]: we obtained functional annotation terms for yeast, human, and fly networks from the SGD [18], GOA [19], and FlyBase [20] databases respectively and used the GO::TermFinder tool [21] to find statistically significant gene ontology (GO) terms shared by a group of proteins in the motif. A GO term is assumed significant if its corrected p -value using the Bonferoni method is lower than 0.05. A motif with at least one such significant term is considered an enriched

motif.

Table 6.6 shows the biological significance of the motifs selected using different motif weighting methods. For the *minimum p-value* column of this table, for each motif we considered the minimum corrected *p-value* of the matched significant terms. Then we calculated the average of this value for all motifs for each of the weighting methods. Among these weighting methods, *the number of edges* resulted in the lowest average minimum *p-value*. Also *the sum of the edge weights* selected better motifs compared to the *maximum weight spanning tree* according to both measures.

Table 6.6 – Comparison of different motif selection criteria

Weighting method	Average Complex Similarity	Average Minimum P-Value
Maximum weight spanning tree	30.67%	3.2e-4
Sum of edge weights	31.07%	2.9e-4
Minimum edge weight	29.86%	3.5e-4
Number of edges	29.59%	2.4e-4

We also evaluated the biological significance of the returned motifs. Table 6.7 lists the percentage of the motifs that were functionally enriched for different species. The best motifs for each query was selected based on the number of the edges.

Table 6.7 – The percentage of enriched motifs

Network	$2 \leq S < 4$	$4 \leq S < 8$	$8 \leq S < 16$	$16 \leq S $
human	32.7% (118/361)	83.4% (131/157)	96.1% (25/26)	100.0% (9/9)
fly	85.1% (370/435)	100.0% (32/32)	100.0% (6/6)	100.0% (2/2)
yeast	100.0% (407/407)	100.0% (140/140)	100.0% (40/40)	100.0% (4/4)

7. Conclusion

In this paper, we presented an enumerative algorithm named RANGI for finding all motifs of the maximum size in list-colored graphs. RANGI augments subgraph enumeration algorithms with pruning methods and heuristics to reduce the size of the search space. In contrast with Betzler et al.'s randomized algorithm, RANGI finds all motifs of the maximum size. In our experimental evaluation of both algorithms, RANGI was significantly faster than Betzler et al.'s algorithm for finding list-colored motifs. We also devised a parallel version of RANGI for multi-core processor configurations, which achieved an average speed up of higher than 2.5 on a quad-core machine.

A few extensions of the colored motif problem have been discussed in the literature. Betzler et al. discussed the problem of finding *more robust* motifs by requiring 2-connectivity or 2-edge connectivity of the motif [7]. RANGI can be extended to add more constraints on

the motifs: after a subgraph passes the motif validity test, it can be tested for any additional constraints like 2-connectivity. Nevertheless, it may be much more efficient to add specific heuristics to the enumeration part of RANGI to reduce the search space with the presence of these additional constraints. The same applies to other variant of this problem as discussed by Bruckner et al. and Dondi et al. [5,13]; Bruckner et al. and Blin et al. [5,6] considered the extension where the motif could contain uncolored vertices. This extension would increase the size of the search space and would make finding the optimal motifs very difficult. The possibility of enumerative algorithms for finding all motifs for these extensions requires a more detailed study.

Acknowledgements

The authors are grateful to the anonymous referees for their constructive and insightful comments. They also thank Christian Komusiewicz, Florian Sikora and Amir Lakizadeh for helpful discussions and comments.

References

1. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, "Network motifs: simple building blocks of complex networks," *Science* **298**(5594), pp. 824–827, AAAS (2002).
2. V. Lacroix, C.G. Fernandes, M.F. Sagot, "Motif search in graphs: application to metabolic networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **3**(4), pp. 360–368, IEEE (2006).
3. S. Wernicke, F. Rasche, "FANMOD: a tool for fast network motif detection," *Bioinformatics* **22**(9), pp. 1152–1153, Oxford University Press (2006).
4. Z. Razaghi Moghadam Kashani, H. Ahrabian, E. Elahi, A. Nowzari-Dalini, E. Ansari, S. Asadi, S. Mohammadi, F. Schreiber, A. Masoudi-Nejad, "Kavosh: a new algorithm for finding network motifs," *BMC bioinformatics* **10**(318), BioMed Central Ltd (2009).
5. S. Bruckner, F. Hüffner, R. Karp, R. Shamir, R. Sharan, "Topology-free querying of protein interaction networks," pp. 74–89 in *Research in Computational Molecular Biology*, Springer (2009).
6. G. Blin, F. Sikora, S. Vialette, "GraMoFoNe: a Cytoscape plugin for querying motifs without topology in Protein-Protein Interactions networks," pp. 38–43 in *International Conference on Bioinformatics and Computational Biology* (2010).
7. N. Betzler, R. Van Bevern, M. Fellows, C. Komusiewicz, R. Niedermeier, "Parameterized algorithmics for finding connected motifs in biological networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **8**(5), pp. 1296–1308, IEEE (2011).
8. S. Guillemot, F. Sikora, "Finding and Counting Vertex-Colored Subtrees," *Algorithmica*, pp. 1-17, Springer New York. DOI: 10.1007/s00453-011-9600-8.
9. M.R. Fellows, G. Fertin, D. Hermelin, S. Vialette, "Upper and lower bounds for finding connected motifs in vertex-colored graphs," *Journal of Computer and System Sciences* **77**(4),

- pp. 799–811, Elsevier (2011).
10. S. Guillemot, F. Sikora, “Finding and counting vertex-colored subtrees,” *Mathematical Foundations of Computer Science 2010*, pp. 405–416, Springer (2010).
 11. I. Koutis, “Constrained multilinear detection for faster functional motif discovery,” *Information Processing Letters* **112**(22), pp. 889–892, Elsevier (2012).
 12. A. Björklund, P. Kaski, L. Kowalik, “Probably Optimal Graph Motifs,” *ArXiv e-prints* (2012). <http://adsabs.harvard.edu/abs/2012arXiv1209.1082B>.
 13. R. Dondi, G. Fertin, S. Vialette, “Finding approximate and constrained motifs in graphs,” pp. 388–401 in *Combinatorial Pattern Matching*, Springer (2011).
 14. A. Ambalath, R. Balasundaram, C. Rao H, V. Koppula, N. Misra, G. Philip, M. Ramanujan, “On the kernelization complexity of colorful motifs,” *Parameterized and Exact Computation*, pp. 14–25, Springer (2010).
 15. N. Alon, R. Yuster, U. Zwick, “Color-coding,” *Journal of the ACM* **42**(4), pp. 844–856, ACM (1995).
 16. J.E. Hopcroft, R.M. Karp, “A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs,” *SIAM Journal on Computing* **2**(4), pp. 225–231, Society for Industrial and Applied Mathematics (1973).
 17. C. Komusiewicz, M. Sorge, “Finding Dense Subgraphs of Sparse Graphs,” pp. 242–251 in *Parameterized and Exact Computation*, Springer (2012).
 18. *SGD project, Saccharomyces genome database*, <http://downloads.yeastgenome.org/> (2012).
 19. D. Barrell, E. Dimmer, R.P. Huntley, D. Binns, C. O’Donovan, R. Apweiler, “The GOA database in 2009 an integrated Gene Ontology Annotation resource,” *Nucleic acids research* **37**(Suppl 1), pp. D396–D403, Oxford University Press (2009).
 20. S. Tweedie, M. Ashburner, K. Falls, P. Leyland, P. McQuilton, S. Marygold, G. Millburn, D. Osumi-Sutherland, A. Schroeder, R. Seal, H. Zhang, “FlyBase: enhancing Drosophila gene ontology annotations,” *Nucleic acids research* **37**(Suppl 1), pp. D555–D559, Oxford University Press (2009).
 21. E.I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J.M. Cherry, G. Sherlock, “GO::TermFinder—open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes,” *Bioinformatics* **20**(18), pp. 3710–3715, Oxford University Press (2004).

Ali Gholami Rudi received his MSc in Computer Engineering from Tarbiat Modares Univeristy, Tehran, in 2011. Currently he is a PhD candidate in computer engineering at Tarbiat Modares University, Tehran. His main research interests include combinatorics and system software.

Saeed Shahrivari received his MSc in Computer Engineering from Iran University of Science and Technology (IUST) in 2011. He is currently a PhD candidate of Computer Science at Tarbiat Modares University (TMU) working on large scale distributed computing. His main research interest is parallel and distributed computing.

Saeed Jalili received PhD in Computer Science from Bradford University, Bradford, in 1991. In 1992 he joined the School of Electrical and Computer Engineering at Tarbiat Modares University, Tehran, Iran. He is currently an associate professor at Tarbiat Modares University and his main research interests are bioinformatics, security protocol verification, network security, machine learning and software runtime verification.

Zahra Razaghi Moghadam Kashani received an MSc in computer science and a PhD in bioinformatics from the Tehran university. Her main research interests are in bioinformatics algorithms and their evaluations. Her specific field of interests in bioinformatics is biological network analysis and related topics.