Locality-sensitive hashing and biological network alignment

Laura LeGault - University of Wisconsin, Madison

12 May 2008

Abstract

Large biological networks contain much information about the functionality of protein-protein interactions and other macromolecular relationships. There are certain small subsets of these graphs which display similar characteristics that appear more frequently than others; these motifs can be used to uncover key information about the structure of the networks. Existing algorithms use exact identity to determine motifs. Here we instead use locality-sensitive hashing to detect networks which have similar but not necessarily identical topologies, and present results for several higher-order motifs.

See http://pages.cs.wisc.edu/∼legault for source code.

1 Introduction

While much of molecular biology has been concerned with the comparison species-specific variants of protein production (that is, the analysis of DNA variations which cause alterations in transcription and translation - typically through sequence alignment), research has turned in the past few years to analysis of the interactions of proteins and other macromolecules, as for example in [8] for *E. coli* and [9] for *S. cerevisiae*. These biological networks have similar properties to electronic networks, and thus the techniques for processing networks can be employed ([6]).

The method employed in both of these studies centers around, as in several sequence alignment methods, the discovery of small, frequently-occurring blocks of data that have a high degree of similarity. However, the difficulty of space and time efficiency arises when dealing with large biological networks; even the core network of *S. cerevisiae* contains 6459 edges and 2808 vertices [4]. When such problems arise in sequence alignment, one of the methods employed is *locality-sensitive hashing*, or LSH, which takes advantage of the fact that exact matches in sequence data may in fact arise purely by chance, and it is more likely that related sequences are merely similar instead of identical ([3]).

Extending this algorithm to network analysis, LSH will uncover small pieces of the network with highly similar, but not necessarily exact, connectivity. This uncovers potential use particularly when subgraph sizes are large (more than five nodes), as the number of potential unique network topologies increases exponentially in the number of nodes - a three-node network has 13 potential topologies, but a four-node network has 199, and so forth. Thus as the number of potential topologies expands, it becomes more and more useful to consider graphs that have a high degree of similarity, since they may be comprised of smaller, identical subunits, which may characterize similar biological interactions ([1, 7, 8, 9]).

The problem of discovering those subgraph topologies which appear frequently within the graph is defined as follows:

Definition. *Given a directed graph $G = (V, E)$ composed of a set V of vertices and a set E of edges, and a parameter $0 < n \leq |V|$, find all connected directed subgraphs $\{g\}$ of G with λ nodes. Determine the relative frequency of graph topologies by comparing a subset of the defining characteristics of each subgraph g.*

The topologies which occur with high frequency within the graph G are called *motifs*. In *S. cerevisiae*, which we analyze here, the most frequently occurring motif has been previously observed ([9]) to be the transcriptional feed-forward motif - a three-node motif wherein two transcription factors regulate a common target, while one transcription factor also acts as regulation on the other. The next most frequent motif is the ‘co-pointing’ motif, where two transcription factors regulate a common target, but without explicitly regulating each other, and the next a set of two targets regulated by the same transcription factor.

2 Methods

Networks in general present a difficulty not at all faced in sequence alignment: DNA sequences have a definite orientation, moving by convention either *upstream* towards the 5’ (terminal phosphate) end or *downstream* towards the 3’ (terminal hydroxyl) end. Networks, however, do not obey a global orientation, and instead are only locally oriented in the sense of a directed graph. For example, in Figure 1, by relabelling nodes in network B (switching the labels on nodes 0 and 1) or network C (switching the labels on nodes 0 and 2), we can exactly reproduce the connectivity matrix of network A. Some approaches ([2]) deal with this explicitly by reordering the nodes for an alignment; we take a simpler approach.

Dataset

The data used for this analysis was the most recent set of ‘core’ interactions of *S. cerevisiae* from [4] at the date of writing. This data set contains 2808 unique proteins and macromolecules, engaged in 6459 directed interactions. This particular set was chosen over the more recent ‘full’ set of interactions simply on account of size - the full set contains 17,580 directed interactions. Even using the present data, we are only capable of examining subgraphs of a maximum of six vertices when using a hash method, and only a maximum of five vertices when using a full enumeration method.

Finding Subgraphs

Before any processing can be done on the subgraphs of G, the first task is to locate such graphs. By the nature of our dataset, all we know about any given vertex v is that there exists some edge e which connects v to either an ancestor u or a successor w. Thus the problem becomes a matter of graph traversal starting from v and discovering all (if any) connected subgraphs of n vertices containing v. Various efficient algorithms such as Kashtan et al’s sampling algorithm ([5]) or Kuramochi and Karypis’ frequent subgraph discovery algorithm ([6]) begin with a single edge and branch out along connected edges to create a subgraph of appropriate size; in the interest of simplicity, we concern ourselves with nodes only.

To discover all unique subgraphs of size n contained in G, we iterate through a list of the vertices V. All vertices are initially marked *undiscovered.* Then for each node in the network, we execute the following modified recursive breadth-first graph traversal algorithm:

Algorithm 1 Discover all subgraphs of graph G

```
for all Node $\in G$ do
    subgraph[0] $\leftarrow$ Node
    get subgraph containing Node of size $n$ // recursive step
    mark Node complete
    reset all incomplete Nodes to unvisited
end for
```

Locality-Sensitive Hashing

Locality-sensitive hashing (LSH) is an algorithm used to solve high-dimensional computational problems involving similarity. It uses a randomized search to determine objects with a high degree of similarity, but not necessarily exact identity. In our application, we determine two networks s_1 and s_2 of d nodes each

Algorithm 2 Recursive function to complete modified breadth-first search

```
Given: Node, current subgraph
queue ← Node's parents
queue ← Node's children
done ← false
if $n - |\text{current subgraph}| = 1$ then
  done ← true
  for all Node ∈ queue do
    add Node to current subgraph
    add resulting subgraph to list of subgraphs of size $n$
  end for
else
  while done = false do
    find first unvisited node in queue
    if no nodes in queue are unvisited then
      done ← true
    else
      add Node to current subgraph
      get subgraph containing Node of size $n - |\text{current subgraph}|$ // recursive step
      mark Node visited
      remove Node from current subgraph
    end if
  end while
end if
```

to be similar if for some fixed $r < d$, their distance is at most r. In a sequence-based LSH method, this distance is simply the number of single-character differences between the two sequences ([3]). However, due to the aforementioned order-independence of identity between network topologies, we must first develop an appropriate distance metric for comparing multiple networks.

Distance Function

The defining characteristics of an order-independent directed graph are the number of vertices and the location, number, and direction of its edges. Since our algorithm only considers subgraphs of a fixed number of vertices, the critical information is reduced to a problem of describing the edges. Each of our subgraphs is described as a connectivity matrix; for example, in Figure 1, the graphs would be described as

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{1}$$

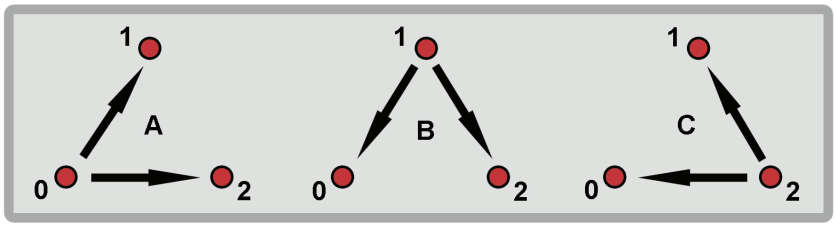

Figure 1: Examples of topologically identical networks with different connectivity matrices (see Equation 1), showing the importance of developing an order-free characterization of networks.

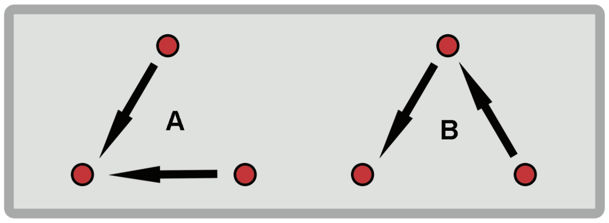

Figure 2: Topologically distinct networks that would be characterized as equal with the unmodified distance function.

As we observed earlier, the topologies for graphs A, B, and C are identical, so the distance function d should map $d(A,B) = 0$, $d(A,C) = 0$, and $d(B,C) = 0$. We observe that a function $d : \mathbb{R}^{n\times n} \to \mathbb{R}$, defined by

$$d(X,Y) = \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{n} (x_{ij}) \right) (10^{n-i}) - \left(\sum_{j=1}^{n} (y_{ij}) \right) (10^{n-i}) \right)$$

could suffice under a proper permutation of the rows j. In this implementation, after summing the rows, we then sort the values in ascending order before multiplying by the power of 10 corresponding to the resulting row number. In effect, this results in simply combining these row numbers of 0, 0, and 2 into $d(A,0) = d(B,0) = d(C,0) = 002$. Since there is only one possible three-node connected graph with two nodes that do not have any outgoing edges whatsoever, this description suffices here. However, what about the case of a directed graph where two nodes each have one outgoing edge?

As exemplified in Figure 2, because of the directed nature of the graphs, number of outgoing edges is not sufficient for unique characterization. We observe, however, that the graphs A and B in Figure 2 differ by number of nodes with no *incoming* edges - A has two such 'source' nodes, while B has only one. Thus we complete our distance function by adding a leading digit $x \times 10^n$, where x is the number of source nodes in the network. (We note here that the number of sink nodes in the network can be determined by the number of rows which sum to zero.) This modified distance metric gives $d(A,0) = 2011$ and $d(B,0) = 1011$, so that $d(A,B)$ is nonzero and the two networks are correctly characterized as distinct.

LSH vs. Full Enumeration

Given this distance metric, we can now determine the definition of similarity used in our LSH method. Where a sequence-alignment LSH method ensures that any two 'similar' sequences differ by at most r characters, our subgraph alignment LSH will ensure that any two 'similar' graphs differ in at most r characteristics - number of source nodes and number of outgoing edges per node. The sorting of numbers of outgoing edges ensures an orientation-independent hash.

Thus for our hash function, we simply choose $(n+1) - r$ positions of the result of the distance function $d(X,0)$ to compare as in sequence alignment LSH. Our implementation chooses positions without replacement, resulting in a consistent, strong hash - allowing replacement introduces the possibility of hashing on the same position multiple times and therefore potentially labelling graphs which differ by more than r positions as similar, though it also increases the probability that graphs which differ by r or slightly fewer than r positions will not be called similar, since it is more likely that we might randomly select one of their differing positions as part of our hash. What we sacrifice in sensitivity, however, we make up in specificity.

The primary advantage of using LSH over full enumeration for our comparison is not only the ability to detect graphs with similar-but-not-exactly-identical connectivity but also the significant savings in space of our hash table. A full enumeration with exact matching will require the full n^{n+1}-space hash table, while a hash on a subset of $(n+1) - r$ positions will require an $n^{(n+1)-r}$-space hash table. Further analysis could diminish this significantly; for example with a three-vertex subgraph, while a full hash table would have $3^4 = 81$ spaces, only 13 would be used: 0012, 0022, 0111, 0112, 0122, 0222, 1002, 1011, 1012, 1111, 1112, and

2011. Hashing on two characteristics lowers the unoptimized number to $3^2 = 9$ (all of which may potentially be used due to the randomized selection of hash positions), a significant difference when compared to the full hash table, and even a small savings when compared to the optimized hash table.

3 Results

Comparing LSH results with full enumeration

Our implementation chose to hash on a number of positions equal to half the number of vertices in the subgraph (rounded up for odd numbers of vertices). Based on this choice, for subgraphs of odd nodes, we hashed on exactly half of the available characteristics. By choosing such a relatively large r for potential differences, the hash instead picked out those motifs which were strikingly different in topology from other motifs and lumped the rest of the topologies together. For example, three of our random choices of positions for a three-vertex subgraph LSH picked out the set of two targets regulated by the same transcription factor while calling most if not all other motifs 'similar'; two other random choices picked out the 'co-pointing motif'; one other random choice picked out the 'feed-forward' motif. While this suggests that these particular motif topologies may have some significance in their uniqueness, it gets us no closer to determining significance since when each of these motifs were singled out, they were much less frequent than the rest of the three-vertex subgraphs by a factor of at least two.

However at higher levels (with more than 13 possible outcomes), we can use the LSH method to determine common characteristics of subgraphs present in biological networks that we might otherwise miss using exact methods. In Tables 1 and 2, we list the results of LSH on subgraphs of five and six vertices. In total, there were 372,076 five-vertex subgraphs and 729,464 six-vertex subgraphs. We observe that nearly 60% ($219,592/372,076 = 0.5902$) of five-vertex subgraphs share the property of having two source nodes with at most two sink nodes and no more than one node with more than two edges coming from it. Also, over 60% ($445,923/729,464 = 0.6113$) of six-vertex nodes have two sink nodes and no more than two nodes with more than two edges coming from them.

Hash	No. of elements	Hash	No. of elements	Hash	No. of elements
220	193,792	211	215,973	121	219,592
120	50,928	111	68,937	111	49,497
210	40,549	221	34,973	131	34,788

Table 1: Five-vertex subgraphs, hashed on positions {5,0,1}, {5,4,3}, and {3,0,4} respectively.

Hash	No. of elements	Hash	No. of elements	Hash	No. of elements
0012	207,077	1001	445,923	2021	288,953
0022	156,807	1101	124,322	3021	133,104
0013	101,202	0001	124,191	3020	59,280

Table 2: Six-vertex subgraphs, hashed on positions {2,1,5,0}, {3,2,1,4}, and {0,1,6,3} respectively.

Analyzing runtime of LSH versus full enumeration

Empirically the runtime difference between an LSH parse and a full parse is small but consistent. Each additional digit of hash introduces a small amount of slowdown, magnified by the size of the dataset. The hashing function, which as described above simply involves multiplying by a power of ten and adding for all hashed positions, is not particularly complex and is completed in $O(m - r)$ time, where $m = n + 1$ is the number of possible characteristics, while the full enumeration takes $O(m)$ time. Since these differ only by a constant, the speedup in this algorithm is not truly significant when used for motif detection. The times (in milliseconds) for finding all subgraphs and hashing the motifs are shown in Tables 3 and 4 below.

3	Subgraph	LSH	4	Subgraph	LSH	5	Subgraph	LSH
	125	110		266	563		562	2250
	141	93		265	594		578	2235
	141	93		266	593		578	2234
	141	109		266	578		578	2266
	156	94		266	578		578	2234
avg	140.8	99.8	avg	265.8	581.2	avg	574.8	2243.8

Table 3: Total time in milliseconds spent for finding subgraphs and performing LSH on three-, four-, and five-vertex subgraphs.

3	Subgraph	Full	4	Subgraph	Full	5	Subgraph	Full
	125	141		250	704		562	2844
	125	125		250	703		579	2828
	141	125		250	703		578	2844
	141	125		266	703		562	2829
	140	125		250	703		578	2828
avg	134.4	128.2	avg	253.2	703.2	avg	571.8	2834.6

Table 4: Total time spent for finding subgraphs and performing full enumeration on three-, four-, and five-vertex subgraphs.

The ratio of LSH processing time to full enumeration processing time is 77.847 for three-vertex subgraphs, 82.651 for four-vertex subgraphs, and 79.157 for five-vertex subgraphs.

4 Discussion - Applications of LSH to Networks

As is evident from the previous section, the use of LSH on network data does not provide a significant speedup in processing, since the bulk of the work is done on operations other than the actual comparison of network topologies. For these decreases in processing time and space, sampling methods such as [5], which randomly samples a relatively small subset of the possible n-vertex graphs in a network and perform exact alignment, or [6], which builds large subgraphs progressively from smaller building blocks (one- or two-edge subgraphs), may be more appropriate.

However this does not mean that LSH is useless in the area of network motif detection. By examining the results and hash positions of the data in Tables 1 and 2, we can draw a number of conclusions about the common characteristics of five- and six-vertex subgraphs in *S. cerevisiae*. As stated in section 3, there are several characteristics which approximately 60% of such subgraphs share, including number of source nodes and similar connectivities. Since the number of possible connected directed graphs with five or six vertices is exponential in the number of vertices, it may be more difficult to do this sort of analysis given only exact alignments of subgraphs.

It is also possible that LSH could be used on other, earlier operations in the network analysis process. Part of the time (and possibly most of the time) spent in performing the LSH or full analysis of the characteristics of each subgraph involves processing the subgraphs to extract these characteristics - creating the connectivity matrix, summing each row, and sorting the rows into ascending order. If it were possible to reduce time spent on any of these operations, the reduction would propagate into a significant speedup considering each operation is performed on every possible subgraph of the original network G.

Indeed most of the effort expended in development was in creating straightforward methods of subgraph discovery and processing to uncover characteristics, rather than applying LSH to the hashing problem. The difficulty in subgraph discovery itself is the caveat that the n-vertex subgraphs must be connected for the motif analysis to have any sort of meaning; while this effectively makes a brute-force approach of enumerating every possible combination of n vertices laughable, it constrains the algorithm to a strict, methodical search

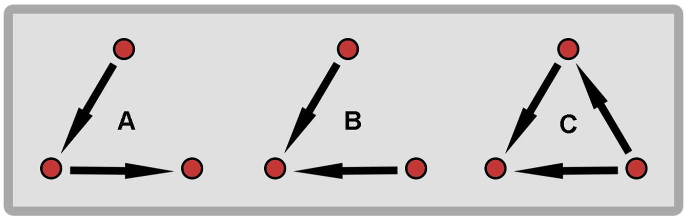

Figure 3: The top three 3-vertex motifs discovered by our exact alignment. (A) Linear, (B) Co-pointing, (C) Feed-forward.

through neighbors. As mentioned above, random sampling of n-vertex subgraphs within the network provide a significant speedup, but what is gained in time is lost (at least in part) in confidence, though as [5] mentions, a small number of samples can estimate even very low concentrations, and various convergence studies can be used to determine the appropriate number of samples.

The problem for future consideration is, then, a matter of applying a randomized sampling method to the operation of processing each subgraph for characteristic extraction. One option would be, instead of summing all rows and then sorting into ascending order, to sum only a subset of the rows and sort these into ascending order, though presumably the time gained by sampling here would be analogous to the time gained in the present algorithm. Or perhaps only a randomly sampled subset of the nodes involved would be considered in the connectivity matrix and subsequent hash, which might result in a slightly better speedup.

Conclusions: Is LSH A Good Idea?

In brief, probably not. In theory its promise of similarity detection and a speedup based on random sampling are attractive features, but the fact of the matter is that network motif detection lacks an operation where an LSH-based approach could provide a significant advantage over an exact alignment. It remains possible that for larger motif sizes, this LSH characterization of motifs could provide insights into common characteristics in these subgraphs, but future work must first deal with the efficient storage of subgraphs and connectivity matrices so that it is possible to process networks of more than six vertices.

Future work should experiment with the creation of other distance metrics, which specifically should require less processing from the basic data structure which stores the subgraph information. By altering this distance metric cleverly, it may be possible to reduce the processing time required without introducing random sampling, and LSH may be more effective in providing a speedup while determining similarities between network motifs.

Supplemental: Motifs

In performing our analysis on the *S. cerevisiae* dataset, we noticed a discrepancy in the relative frequency of motifs detected by our methods and those of Zhang et al ([9]). While they discovered primarily feed-forward motifs (Figure 3(C)), followed by ‘co-pointing’ motifs (Figure 3(B)) and dual-target motifs, our analysis of the core *S. cerevisiae* dataset uncovered 15,839 simple linear motifs (Figure 3(A)), 10,154 ‘co-pointing’ motifs, and 6,092 feed-forward motifs; the dual-target motif did not make the top three most frequent motifs with a frequency of only 4,606. That is, of 36,691 three-vertex subgraphs, 43.17% were linear, 27.67% were ‘co-pointing’, and 16.60% were feed-forward. Zhang et al, however, discovered 7.4×10^2 total ‘co-pointing’ motifs, as compared to approximately 5.0×10^2 feed-forward motifs and 11.3×10^3 dual-target motifs of various types, a notable relative difference especially given that in our ‘core’ dataset the dual-target motif was so scarce.

References

[1] Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews: Genetics*, 8, 2007.

[2] Johannes Berg and Michael Lassig. Local graph alignment and motif search in biological networks. *PNAS*, 101(41), 2004.

[3] Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5), 2001.

[4] David Eisenberg. Database of interacting proteins - s. cerevisiae core, tab-delimited, October 2007. `http://dip.doe-mbi.ucla.edu/dip/File.cgi?FN=ScereCR20071007.tab`.

[5] N. Kashtan, S. Itzkovitz, R. Milo, and U Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 2004.

[6] Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9), 2004.

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

[8] S. Shen-Orr, R. Milo, S. Mangan, and U Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31, 2002.

[9] L. Zhang, O. King, S. Wong, D. Goldberg, A. Tong, G. Lesage, B. Andrews, H. Bussey, C. Boone, and F Roth. Motifs, themes and thematic maps of an integrated saccharomyces cerevisiae interaction network. *Journal of Biology*, 4(6), 2005.