A Report On
Construction of
Interaction Networks
between Lists of Molecules
&
Automatic Reconstruction
of Signaling Pathways

S.Vikas
Strand Life Sciences, Bangalore
May-July 2007
## Contents

1 Introduction to the software 3

2 Problem 1 (List Interaction Networks) 3
   2.1 Problem Statement ........................................ 3
   2.2 Introduction .............................................. 4
   2.3 Remarks ................................................... 5
   2.4 Algorithm 1 .............................................. 5
   2.5 Algorithm 2 .............................................. 6
   2.6 Remarks ................................................... 7
   2.7 Conclusion ................................................ 7
   2.8 Future Work .............................................. 7

3 Problem 2 (Automatic recognition of signaling pathways) 7
   3.1 Problem Statement ........................................ 7
   3.2 Remarks ................................................... 8
   3.3 General Algorithm ....................................... 8
   3.4 Algorithm 1 .............................................. 8
   3.5 Remarks ................................................... 8
   3.6 Algorithm 2 .............................................. 9
      3.6.1 Data Structures Used .............................. 10
      3.6.2 Details of Implementation ....................... 10
   3.7 Remarks ................................................... 11
   3.8 Experimental Results of Algorithm 2 ............... 12
      3.8.1 Motifs Found ..................................... 12
   3.9 Some statistics about the results of the experiment 14
   3.10 Inferences from the Experiment ..................... 17
   3.11 Conclusions and Future Work ....................... 17
1 Introduction to the software

PathwayArchitect is a software designed to assist researchers in doing an in-depth analysis of biological networks and extracting useful information out of them.

The software has, at its core, a database of 400,000 biological entities and about 1.6 million interactions existing between these entities. The database has been built mainly by using an in-house text mining tool to parse abstracts of a large number of research papers that are available online at standard websites like PubMed, BioMedCentral etc. The database is updated periodically. Also, interactions from standard databases like BIND, MINT, IntAct have been incorporated into the database. Users also have the option of running the text-mining tool on custom documents and adding the interactions thus found to the database.

Each entity(either a node or an interaction) has got a unique id. This is used as a handle to access that entity during the course of our analysis.

In short, the whole database can be treated as one huge graph with the biological entities as the nodes and the interactions of the graph as the edges. Also, each interaction has a weight which is the reference count of that interaction. Reference count of an interaction is literally a count of the number of references from which that interaction has been obtained during the course of building of the database. E.g., if interaction X has been found at 8 places, then its reference count is 8.

Also, each interaction can be identified by its 'signature', which is a string containing information about the interaction(e.g., mechanism) and the ids of the interacting entities. Therefore, given a id of an interaction, you can get its signature which gives you all the information you need about the interaction.

The software also has a user-interface for querying and visualising the database contents as graphs. The software also has tools for relevance network analysis and advanced network modeling.

The tool currently has five automatic graph construction methods.

The relevant ones are:

1. **Expand Nodes Network** - This feature enables you to find all the controls that a set of nodes are adjacent to.

2. **Direct Interaction Network** - This feature enables you to find all the controls that link the nodes of the chosen set with themselves.

These methods form the basis for any advanced network analysis that is required.

2 Problem 1 (List Interaction Networks)

2.1 Problem Statement

Given 2 lists L1 and L2, find all the one-hop interactions connecting nodes in them.
2.2 Introduction

A list of nodes is defined as just a data set of molecules/genes. Lists are usually used to represent groups of genes that work together or have some common functionality. E.g, we could have a nuclear receptor list that contains all the genes that are known to function as nuclear receptors. We could also have a list of genes that function together, for instance in a signaling pathway that activates the transcription of some gene downstream.

Therefore, this problem is highly relevant in the sense that given 2 lists of nodes, it might be important to find out how closely the two lists correlate and how close their functionalities are. E.g, if there are lists of two pathways which have some common transcription factors that are activated in the pathways, then it might be useful to see how well the 2 pathways relate and draw some useful conclusions from the analysis.

Also, another important context where this analysis could be useful is when you get a set of genes from the gene expression analysis which you think are part of a common functionality. E.g, you might get a set of genes from analysis of a breast cancer sample which are differentially expressed in the context of breast cancer. Then you might want to know how well it correlates with a set of genes that you got, say from analysis of a cancer sample of another organ. Then, this would be a very good starting point on which you can build your
2.3 Remarks

Algorithmically speaking, although the problem seems pretty straightforward, complications can arise if there are common nodes in the 2 lists. So, if we just get all controls coming out of L1 and L2 and find the intersection, then we can be in trouble because that could give a whole lot of extra wrong controls not connecting the 2 lists at all.

Therefore, care has to be taken to ensure that these wrong controls are eliminated.

2.4 Algorithm 1

1. Find all the interactions which the entities of List 1(L1) are participating in. Let the set of these controls be C1.

2. Find C2 in a similar way from List 2(L2).

3. C = intersection (C1, C2) (Now you have all controls adjacent to both L1 and L2).
4. Get all the signatures of $C$ from the database.

5. For each control, check whether one node of the control is in $L_1$ and another in $L_2$ the information of which you can get from its signature.

6. If this is true, then it’s a valid control.

7. Get all the valid controls in this way.

8. Pass the union of $L_1$, $L_2$ and valid controls and generate the final graph from it.

Figure 3: Depiction of the concept involved in Algorithm 2. Controls are classified into 4 categories as shown and dealt with separately to avoid wrong controls.

### 2.5 Algorithm 2

1. Let $D_1$ = set difference of $L_1$ and $L_2$, ie $D_1 = L_1 - L_2$.

2. Similarly, $D_2 = L_2 - L_1$.

3. $I$ = set intersection of $L_1$ and $L_2$, ie $I = L_1 \cap L_2$.

4. Then find common controls between $L_1$ and $D_2$ ($= C$) using the method described in Algorithm 1 (steps 1 to 3).
5. Similarly, find all common controls between D1 and I (= C1).
6. Also, find the controls from the direct Interaction network of I. (= C2).
7. The union of all these controls C, C1 and C2 gives the required controls.
8. Using L1, L2 and relevant interactions, construct the resultant graph which is the result.

2.6 Remarks

- It has been observed during the testing phase that in Algorithm 1, it is the signatures finding that takes up the bulk of the time and which is the rate determining step. Therefore, for fast access to the signatures, it is imperative that they be present in the database (which is true of Mammal2 DB). Therefore, this algorithm is very good for organisms that have a signature table already built in the database.

- On the other hand, Algorithm 2 is very good when signature tables aren’t available and is a good alternative option to Algo1. The catch is this calls the ExpandNodes algorithm 5 times as opposed to twice in the case of Algo1. But, it is also important to note that Algo2 calls ExpandNodes on smaller sets, thus potentially taking lesser time than you would expect with 5 ExpandNodes calls.

- Therefore, we can do a check with the config file of the database and check whether a signature table for that organism is present or not. If it is, then we can use algo1, otherwise going with algo2.

2.7 Conclusion

This algorithm helps you to connect 2 lists of nodes and find out the extent of correlation between the two lists. The 2 lists being heavily connected implies that there is a lot of relation between the entities present in the 2 lists and vice versa.

This can be applied to the lists that you import from elsewhere, such as the GSEA Lists (Lists which give details of the molecules participating in a particular pathway or a reaction sequence).

2.8 Future Work

As of now, the algorithm seems rather complete and robust. Probably some boundary-checks might be required to make it even more flexible and fault-tolerant.

3 Problem 2 (Automatic recognition of signaling pathways)

3.1 Problem Statement

Given a signaling molecule (receptor, txn factor etc.), the aim is to construct a biologically relevant signaling pathway starting from the given molecule.
3.2 Remarks

This is an example of a problem that doesn’t have a trivial deterministic algorithm that can be guaranteed to work at all times. This is a problem where the algorithm is supposed to intelligently choose from a large choice of nodes, the node that is most probably part of the pathway. So, the most important thing to remember is that there is always a degree of uncertainty associated with the solution suggested by the algorithm.

Also, currently, in the DB, there is a reference count associated with each interaction which is a measure of how well-studied is that interaction (in other words, how many references does that control have in existing literature). This is the metric that we use to gauge the reliability of the control.

Initially, it was assumed that there is a straightforward proportionality between the refCount of a control and the probability with which it is present in the pathway, but later we came up with a more reliable and logical way to relate the refCount and the probability as will be explained in Algo2.

3.3 General Algorithm

Here, we will demonstrate the algorithm for a given txn factor, growing the pathway backward towards a receptor. However, this can easily be extended starting from a receptor by growing the scaffold forward from the receptor towards a transcription factor (just by modifying a parameter in a config file, to be precise).

3.4 Algorithm 1

1. Assume that there are N nodes found so far during the execution of the algorithm. The aim is now to find the (N+1)th node. Also, assume that we have with us a priority queue with all the controls that expand from the subgraph ordered by their refCounts (i.e., higher the refCount, higher up is the control and better the control).

2. Find all the controls expanding from the newly found node (i.e., the Nth node) and add them to the heap according to their refCounts.

3. Now, find the control connected to the subgraph found so far with the highest refCount. This can be achieved in constant time by a simple ‘extract-max’ operation on the heap (the primary reason why the heap was chosen as the data structure).

4. If the newly found node already belongs to the subgraph, then ignore that control and go back to step 3.

5. Once you get the new node (the N+1th), add that node and the control to the graph and repeat the algorithm until the entity belonging to the desired group (either receptors or TFs) is found.

3.5 Remarks

- On careful analysis of the algorithm, we see that there is absolutely no measure of relevance being considered when we choose the next node of
the graph. That is to say, at the $i$th node, we just consider its reference count with the $(i-1)$th node, without considering about how well it relates to the subgraph found so far.

- Also, a small change that was made to the algorithm was that instead of just considering the refCount of a control between 2 nodes, it’s more relevant to consider the sum of refCounts for all interactions between a node of the subgraph and the candidate node. E.g. if there are 5 controls between A and B, it’s better to consider the sum of all refCounts between A and B rather than choosing a control with the highest refCount. The Heap is appropriately modified to suit this change.

- However, results that were obtained were not very satisfactory since again, we aren’t factoring any relevance into the algorithm. Therefore, Algo2 was adopted to include this factor.

### 3.6 Algorithm 2

This algorithm to a large extent factors the relevance metric into consideration. Let the subgraph found so far be $G$ and the latest node be $V$. Now, we find
all the binary protein-protein interactions involving V which are in the right
direction.

Now, for each new prospective node that appears, we find all controls that
it’s being linked to with the subgraph as a whole (unlike just the latest node as
in the previous algorithm). Then among all the ‘candidate’ nodes found so far
which are linked to G, the node that has the highest total refCount with the
subgraph is chosen.

3.6.1 Data Structures Used
1. globalExpandNodesMap - map that contains mappings from every node
connected to the subgraph externally to an array of all the interactions
which it has with the subgraph.
2. globalRefCountMap - map that contains mappings from every control to
its refCount.
3. graphEdges - set containing all edges present in the subgraph found so far.
4. graphVertices - set containing all vertices present in the subgraph found
so far.
5. totalRefCountMap - mapping from each nodeId to the sum of all refCounts
of the controls that is linking it with the subgraph so far.

3.6.2 Details of Implementation
1. At the nth step of the algorithm, let the subgraph be G and the latest
node found be V.
2. Appropriately find all valid controls (i.e., all binary protein-protein con-
trols in the appropriate direction) involving V.
3. Get the other participant of each of these controls (one participant obvi-
ously being V anyway). Store the mappings from each new participant to
the control in a local otherParticipantMap.
4. Update the globalRefCountMap by querying for the refCounts of all the
new controls.
5. Update the globalExpandNodesMap for each node present in otherPartic-
ipantMap (i.e., if there isn’t a mapping for a node in otherParticipantMap
in the globalExpandNodesMap, then add the mapping ; otherwise just
append the new set of controls to the array in the global map).
6. Update the totalRefCountMap.
7. Find the node with the maximum total reference count with the sub-
graph. Add the nodeId to graphVertices. Add the controls (from global-
ExpandNodesMap) to graphEdges. Remove the mappings from the global
Maps to prevent these objectIds from coming up again as prospective
nodes.
8. Check if the terminating condition (maybe condition is to find a receptor or atxn factor or even until a particular pathway size is reached).

9. Otherwise, go back to step 1.

3.7 Remarks

As you can see, this not only looks at the refCount of the candidate node with the previous node, but with the subgraph as a whole.

This gives better results because suppose there is a sentence 'Protein A phosphorylates protein B, thus activating TF C', the NLP captures not only A-B, B-C but also A-C since A is indirectly regulating the expression of C. Therefore, since such a sentence is relevant in the context of signaling pathways and does occur quite often, we can exploit the presence of such interactions.

Also, apart from this, another reason why this is giving better results is that we are giving more preference to nodes which have interactions with the graph as a whole and not just a single node. Thus, this does encourage the formation of 'clusters' (in the best case, cliques) which have a higher chance of being biologically relevant than others.
3.8 Experimental Results of Algorithm 2

We generated small scaffolds (mini-pathways) of size 4 working backwards from TFs. We did this primarily because TFs are more specific than receptors and thus on an average, have much lesser connectivity.

We did this for about 60 TFs and a detailed analysis was done on each of these scaffolds and they were given a score out of 5 in the following way:

- Interactions correct and biological context perfect - 5/5.
- Interactions correct and biological context partially right (and the txn factor being part of the interpreted context). - 4/5
- Interactions more or less right but context not completely clear (or the TF not part of the context of the rest of the scaffold) - 3/5.
- Interactions not right and context also not captured - 2/5.
- A serious NLP or Tagging error - 1/5.

3.8.1 Motifs Found

1. **Clique** - A complete graph. Generally very reliable since the nodes are so highly inter-connected.
Figure 7: Example of a Scaffold with a 'Triangle+Line' motif
2. *Triangle + Line* - Slightly less connected as compared to the clique, but as per our results a very good motif nevertheless.

![Figure 8: Example of a Scaffold with a '2-Triangle' motif](image)

3. *2 Triangles* - Well connected, but not giving as good results as a) and b).

4. *Path* - Not a great motif, no cluster formation observed, therefore not too encouraging.

### 3.9 Some statistics about the results of the experiment

- About 21% of the scaffolds received a score of 5/5.
- About 28% received 4/5.
- About 40% received 3/5.
- About 5% received 2/5.
- The rest (5%) received 1/5.
Figure 9: Example of a Scaffold with a 'Path' motif
Figure 10: Example of a Clique-motif scaffold that got a score of 3/5. As you can see, the nodes are not highly interconnected thus suggesting that not only the kind of motif, but also the density of interactions determine the strength of the pathway
3.10 Inferences from the Experiment

The scoring pattern of the scaffolds indicate that a bulk of the scaffolds have scored reasonably well, despite a conservative scoring scheme.

Also, we see that the clique motif, contrary to our expectations, hasn’t performed the best. This is perhaps due to the fact that although the motif was a clique, it wasn’t as densely connected as some of the Triangle+Line scaffolds.

3.11 Conclusions and Future Work

Firstly, since all these scaffolds are accompanied with a degree of uncertainty, a p-value (a degree of ‘niceness’) can be assigned to the scaffold as a function of its motif, no. of interactions and other relevant parameters.

Also, one other factor that hasn’t been included when we are finding the nth node is the number of nodes of the subgraph that it is interacting with. All we are considering is the total refCount of all its interactions with the subgraph as a whole. For e.g., if a node is extremely well connected with just 1-2 nodes of a graph, then although it might not be relevant to the context of the pathway, it gets selected. Therefore, to further factor in the relevance metric, it might be useful to include this feature in the algorithm.