

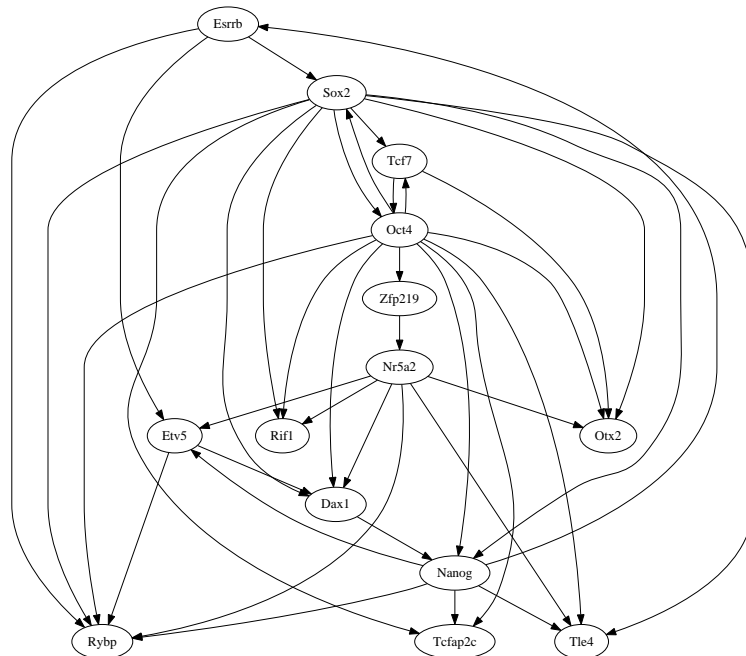
MS6a, Exercises Week 3, Model Solution

Rune Lyngsø

November 2, 2009

A Strongly Connected Components

1. Find the strongly connected components in the following regulatory network, by running the strongly connected component algorithm in the lecture notes. For each gene, include the finishing time for the corresponding node in your answer, and indicate which gene leads to the discovery of each strongly connected component. This network is a subnetwork of the network presented in [1, Fig. 3], with two regulatory interactions changed to have opposite direction ($Zfp219 \rightarrow Nr5a2$ and $Dax1 \rightarrow Nanog$) and one interaction removed ($Nanog \rightarrow Oct4$).



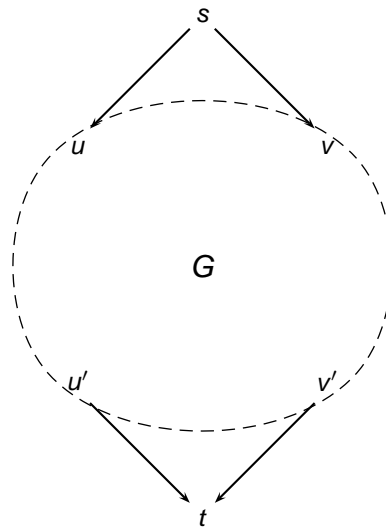
Starting the finishing time depth-first search from the Esrrb gene at the top of the network, for one choice of order in which children are visited we get the following annotation of the network with finishing times:



Hence, the second depth-first search, traversing edges in their opposite directions will also start with the Esrrb gene. This search will eventually visit genes Esrrb, Nanog, Sox2, Oct4, Tcf7, Dax1, Etv5, Nr5a2, and Zfp219, which are therefore one strongly connected component. The remaining genes only have incoming edges from genes in this strongly connected component, so each will not be visited until we encounter it in the **for** loop over nodes in order of decreasing finishing times. Hence, each of Otx2, Rif1, Tle4, Tcfap2c, and Rybp will generate a strongly connected component containing just itself.

2. The equivalent to strongly connected components for an undirected graph can be said to be *biconnected components*. Two edges are biconnected if they lie on a common simple cycle (i.e. a path starting and ending in the same node, but otherwise consisting of distinct nodes). A biconnected component cannot become disconnected by deletion of a single node or edge. Note that a node can be incident to edges in different biconnected components. Describe an algorithm that finds all biconnected components by a liberal use of maximum flow problem solutions.

If two edges, (u, v) and (u', v') , lie on a common simple cycle, then there must be node disjoint paths connecting either u with u' and v with v' or u with v' and v with u' . It follows that if we define a flow network as



where the area enclosed by the dashed line and marked G represents the connectivity in G with flow allowed in both directions of an edge, then the maximum flow when capacities are on nodes and each is allowed a flow of 1 (see solution to problem 6 for how to do this) is 2 iff (u, v) and (u', v') are biconnected. This allows us to test biconnectedness of a pair of edges, so from here the following algorithm should do the trick: Using an augmenting path algorithm, e.g. Edmonds-Karp, each maximum flow problem

Algorithm 1 Flow based biconnected components

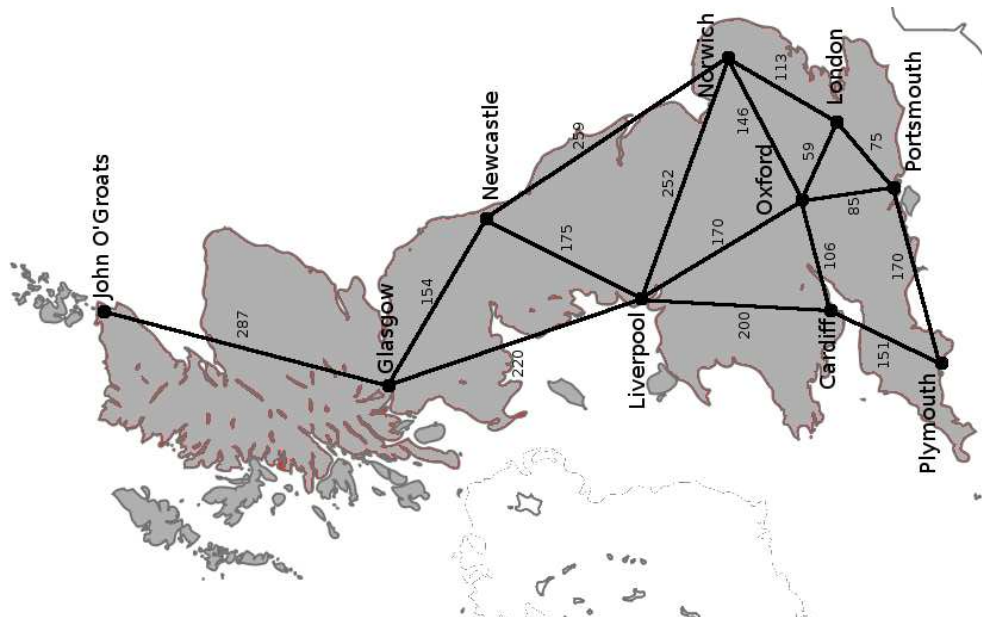
```

for  $e \in E$  do
  if  $e$  has not yet been assigned to a biconnected component then
    Initialise new biconnected component with  $e$ 
    for all edges  $e'$  not yet assigned to a biconnected component do
      if  $e$  and  $e'$  are biconnected then
        Add  $e'$  to  $e$ 's biconnected component
  
```

can be solved in time $O(|E|)$ (as only at most two augmenting paths will be found), so using this method we can compute all biconnected components in time $O(|E|^2)$. Biconnected components can actually be found in time $O(|E|)$.

B Maximum Flows

3. In the road network from problem 1 on last week's set of exercises,



assume that edge weights are capacities, and that there are edges with this capacity in both directions of the edge. What is the maximum flow from Newcastle to Plymouth? What is the corresponding minimum capacity cut?

Though repeatedly choosing an augmenting path and updating the flow according to its capacity is not guaranteed to ever terminate, as we shall shortly see, in this case the easiest approach is just to choose the following three augmenting paths:

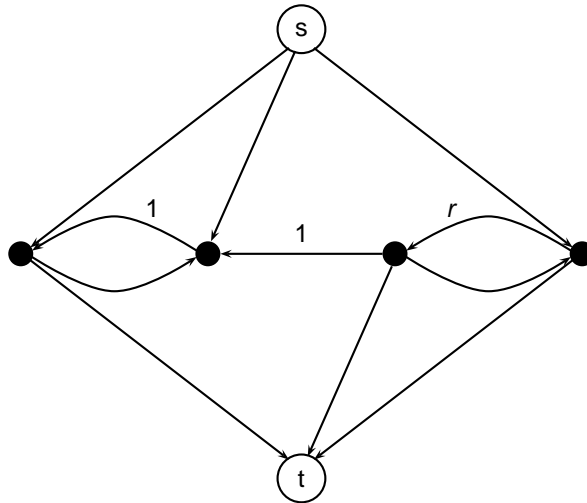
151: Newcastle \rightarrow Liverpool \rightarrow Cardiff \rightarrow Plymouth

85: Newcastle \rightarrow Norwich \rightarrow Oxford \rightarrow Portsmouth \rightarrow Plymouth

75: Newcastle \rightarrow Norwich \rightarrow London \rightarrow Portsmouth \rightarrow Plymouth

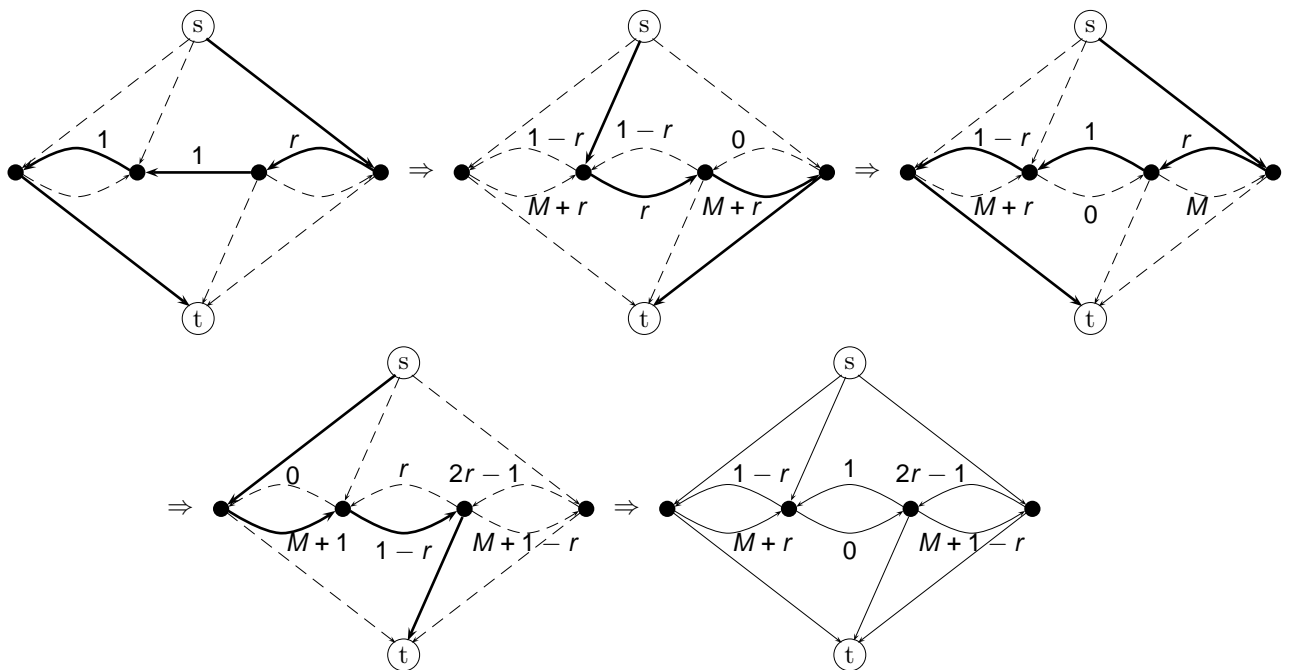
After updating the flow along these paths, one can observe that all connections from the rest of Britain to Portsmouth and Plymouth are fully saturated, so the flow is maximal and the minimum cut puts Plymouth and Portsmouth on one side and the remaining eight places on the other side. The value of the flow is 311.

4. Consider the flow network



with all edges, except for the three annotated with capacities 1, 1, and r , having capacity M for some suitably large value of M . Further assume that r has been chosen such that $r = \frac{1}{1+r}$, i.e. r is the golden ratio.

Assume that we make four flow updates with the maximum capacity of the following augmenting paths (paths shown in bold, only horizontal back edges in the residual network shown):



What are the residual capacities of the three edges with original capacities 1, 1, and r ?

From left to right, residual capacities are $1-r$, 1, and $2r-1$.

Show that $1-r = r^2$ and $2r-1 = r^3$.

Using $r = \frac{1}{1+r}$ we get

$$1-r = 1 - \frac{1}{1+r} = \frac{r}{1+r} = r^2$$

and using this we get

$$2r - 1 = r - (1 - r) = r - r^2 = r(1 - r) = r^3$$

Argue that repeating this chain of augmenting path updates will never generate a maximum flow. What is the maximum flow of the network?

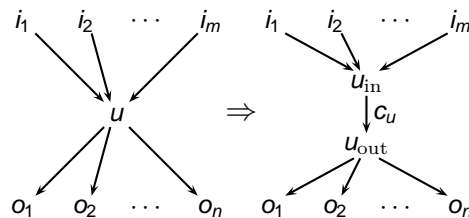
After the four updates, we are in the same situation as before, except that residual capacities of the leftmost edge with initial capacity 1 and the edge with initial capacity r have each dropped by a factor of r^2 . We can thus keep applying this sequence of updates without any of the edge capacities becoming 0, though they will become arbitrarily small. Hence, there will always be an augmenting path, so we never reach a maximum flow. In fact, the flow will converge to $2 \sum_{i=0}^{\infty} (r^2)^i = \frac{2}{1-r^2} \approx 3.2$, while the maximum flow is $2M$ (augment along two flanking paths).

5. Assume $G = (V, E)$ describes a metabolic network, with V being the set of compounds and directed edges E being the set of reactions converting a reactant into a product. How can you determine the minimum number of reactions that need to be disabled before we can no longer produce a target product $t \in V$ from initial metabolite $s \in V$?

Just set capacities of all edges to 1, and compute the maximum flow in the resulting flow network. If there are k reactions/edges whose removal will disconnect s from t , then there is an s, t cut of capacity at most k , and if there is an s, t cut of capacity at most k then we can break any reaction path from s to t by removing the reactions/edges crossing this cut in the forward direction.

6. Assume $G = (V, E)$ describes a regulatory network, with V being the set of genes and directed edges E indicating that one gene regulates another. How can you determine the minimum number of genes that need to be deleted for a gene $s \in V$ not to have any regulatory effect on another gene $t \in V$?

This is very much like the previous problem, except that we need to find the maximum number of node disjoint paths, rather than the maximum number of edge disjoint paths, connecting s and t . And so far only edges have had capacity constraints in our flow networks. It is however easy to modify a network with node capacities to a network where only edges have capacity restrictions, as illustrated below.



If node u with capacity c_u has m incoming edges and n outgoing edges, we split u into an incoming side, u_{in} , and an outgoing side, u_{out} , connect the incoming edges to u_{in} and the outgoing edges from u_{out} and finally connect u_{in} to u_{out} with an edge with capacity c_u . The flow through u_{in} and u_{out} is limited by their connecting edge which has capacity c_u . We can now determine the minimum number of genes that need to be deleted by computing the maximum flow in the modified network.

7. Consider the alternative splicing graph of problem 4 of last week's exercise,

$0 \rightarrow \{1, 3, 24\}$	$1 \rightarrow \{2, 4\}$	$2 \rightarrow \{5\}$	$3 \rightarrow \{4\}$	$4 \rightarrow \{5\}$
$5 \rightarrow \{6, 7, 8, 11\}$	$6 \rightarrow \{7\}$	$7 \rightarrow \{8\}$	$8 \rightarrow \{9, 11\}$	$9 \rightarrow \{10, 11\}$
$10 \rightarrow \{11\}$	$11 \rightarrow \{12, 13, 16\}$	$12 \rightarrow \{13, 14, 15\}$	$13 \rightarrow \{14\}$	$14 \rightarrow \{15\}$
$15 \rightarrow \{16\}$	$16 \rightarrow \{17, 18, 20\}$	$17 \rightarrow \{18\}$	$18 \rightarrow \{19, 20, 21, 22, 39\}$	$19 \rightarrow \{20\}$
$20 \rightarrow \{21\}$	$21 \rightarrow \{22, 23, 29\}$	$22 \rightarrow \{23\}$	$23 \rightarrow \{25, 26, 32\}$	$24 \rightarrow \{25\}$
$25 \rightarrow \{26, 28\}$	$26 \rightarrow \{27, 31, 38\}$	$27 \rightarrow \{28\}$	$28 \rightarrow \{29\}$	$29 \rightarrow \{30, 31\}$
$30 \rightarrow \{31\}$	$31 \rightarrow \{32\}$	$32 \rightarrow \{33, 34, 35\}$	$33 \rightarrow \{35\}$	$34 \rightarrow \{35\}$
$35 \rightarrow \{36, 37, 39\}$	$36 \rightarrow \{37\}$	$37 \rightarrow \{38\}$	$38 \rightarrow \{39\}$	$39 \rightarrow \{40\}$
$40 \rightarrow \{41, 42\}$	$41 \rightarrow \{42\}$	$42 \rightarrow \{43\}$	$43 \rightarrow \emptyset$	

where for each node the set of nodes it has an edge to is listed above. As discussed, a path through this graph corresponds to a spliceform that can be observed, i.e. a possible transcript we may sequence. Hence, the minimum number of transcripts we need to sequence before we can have any hope of having seen all edges in the alternative splice graph is equal to the minimum number of paths required to make sure that every edge is in at least one path.

An alternative splicing graph is always acyclic. Given such a graph $G = (V, E)$, define a flow network $G' = (V', E', w')$ with

$$V' = \{x_u\}_{u \in V} \cup \{y_u\}_{u \in V} \cup \{s, t\}$$

$$E' = \{(s, x_u)\}_{u \in V} \cup \{(y_u, t)\}_{u \in V} \cup \{(x_u, y_v) \mid (u, v) \in E\}$$

and all edge capacities set to 1. What is the relationship between the minimum number of paths needed to cover G , and the maximum flow in G' ?

Let f be a maximum (and integral) flow in G' . For every u with flow 1 through x_u , define $t_f(u)$ to be the node v such that $f(x_u, y_v) = 1$ – for an integral flow, this is well defined. Now for every u with flow through x_u of 1 and flow through y_u of 0 define a path $\pi_u = u \rightarrow t_f(u) \rightarrow t_f(t_f(u)) \rightarrow \dots \rightarrow t_f^k(u)$ where k is chosen maximally, i.e. the flow through $x_{t_f^k(u)}$ is 0. It can be seen that any node u with a flow of 1 through at least one of x_u and y_u will be on one such path. For every node u with flow 0 through both x_u and y_u , define $\pi_u = u$. Clearly the collection of paths will cover all nodes of G and be node disjoint. The path cover will contain exactly one path for each u with flow 0 through x_u . Hence, the size of the path cover is $|V| - |f|$.

Conversely, assume that we have a node disjoint path cover of size k . We can then send flow 1 through all paths $s \rightarrow x_u \rightarrow y_v \rightarrow t$ for all (u, v) in one of the paths in the path cover. A path will cover one more nodes than it has edges, so the value of the flow is $|V| - k$. So a maximum flow in G' corresponds to a minimum node disjoint path cover of nodes in G .

However, we are not requiring paths to be disjoint when determining the minimum number of transcripts needed to observe an alternative splicing graph – the same edge can be used more than once. To address this problem, we will instead compute the flow in $G'' = (V', E'', w'')$ where there is an edge from x_u to y_v for all $u \neq v$ with a path from u to v . When converting a flow to a path, some of the edges will correspond to paths of more than one edge. These edges can just be added in, and we will still have a path cover of all nodes, possibly with some nodes covered by more than one path. For a path cover of all nodes, we just start defining flows for the first path, and for all consecutive paths use the (x_u, y_v) edges corresponding to paths bypassing the nodes already used for previous paths. To have every path start from the first node and end at the last node of the alternative splicing graph, we can also just add extra edges which may at most result in some nodes being even more covered.

Finally, what we really want to cover are edges and not nodes. Instead of creating the flow network from \mathcal{G} , consider the directed line graph $L = (V_L, E_L)$ of \mathcal{G} : the nodes of L are $V_L = E$, i.e. the edges of \mathcal{G} , and two nodes (u, v) and (u', v') are connected by a directed edge in E_L iff $v = u'$, i.e. if the target of (u, v) equals the source of (u', v') . It is an easy observation that a path in L corresponds to a path in \mathcal{G} . Hence, we can apply the maximum flow approach outlined above to find a minimum path cover of the nodes in L , and this will be a minimum path cover of the edges in \mathcal{G} .

Quite interestingly, the size of a maximum path cover of the edges, i.e. a set of paths covering all edges but where removal of any path leaves at least one edge uncovered, can be computed simply as $1 + \sum_{v \neq t} (\text{out-degree}(v) - 1)$ where t is the terminal node which is assumed to be the only node with out degree 0.

What is the minimum number of transcripts that we need to sequence to detect all edges in the alternative splicing graph above?

Applying this method, a better method, or eye balling the graph, it can be established that the minimum number of transcripts required to observe the entire alternative splicing model is 7. That the path cover cannot be smaller can be seen from the fact that any pair of edges from $0 \rightarrow 24$, $16 \rightarrow 20$, $18 \rightarrow 19$, $18 \rightarrow 20$, $18 \rightarrow 21$, $18 \rightarrow 22$, and $18 \rightarrow 39$ cannot be part of the same path. Seven paths covering all edges are

- $\pi_1 : 0, 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43$
- $\pi_2 : 0, 3, 4, 5, 7, 8, 12, 14, 15, 16, 18, 20, 21, 23, 25, 27, 28, 29, 31, 32, 34, 35, 37, 38, 39, 40, 42, 43$
- $\pi_3 : 0, 24, 25, 26, 31, 32, 35, 39, 40, 42, 43$
- $\pi_4 : 0, 1, 4, 5, 8, 9, 11, 13, 14, 15, 16, 20, 21, 29, 31, 32, 35, 39, 40, 42, 43$
- $\pi_5 : 0, 3, 4, 5, 11, 16, 18, 21, 23, 26, 38, 39, 40, 42, 43$
- $\pi_6 : 0, 3, 4, 5, 11, 12, 15, 16, 18, 22, 23, 26, 38, 39, 40, 42, 43$
- $\pi_7 : 0, 3, 4, 5, 11, 16, 18, 39, 40, 42, 43$

Assume that we have an alternative splicing model where for each node we have a probability over the outgoing edges. Can you describe a recursion allowing us to compute the probability that all edges are observed if we draw k transcripts from the splicing model? (hint: recurse on nodes according to the natural ordering, and keep track of the number of partial paths currently terminating at each node up to the current node).

Let $p_k(n_1, n_2, \dots, n_i)$ denote the probability that all edges ending in nodes 1 through i have been observed and that we still need to extend n_j partial paths currently terminated at node j for $1 \leq j \leq i$. Obviously this quantity is only of interest for $\sum_{j=1}^i n_j = k$. It will further be zero if $n_j > 0$ and node j has no edges ending in a node later than i , or if n_j is less than the number of nodes later than i that node j has an edge to. Otherwise, when we reach node i we need to extend at least one of the partial paths terminating at a node with an edge to node i . If the nodes with an edge to node i are j_1, j_2, \dots, j_m , $p(j \rightarrow i)$ the probability of choosing the edge to node i at node j , and $p(j \rightarrow \geq i)$ the probability of choosing an edge to a node not before node i at node j , we get a recursion like

$$p_k(n_1, \dots, n_i) = \sum_{a_{j_1}=1}^{n_{j_1}} \binom{n_{j_1} + a_{j_1}}{a_{j_1}} \left(\frac{p(j_1 \rightarrow i)}{p(j_1 \rightarrow \geq i)} \right)^{a_{j_1}} \left(\frac{p(j_1 \rightarrow \geq i+1)}{p(j_1 \rightarrow \geq i)} \right)^{n_{j_1}} \dots$$

$$\sum_{a_{j_m}=1}^{n_{j_m} - a_{j_1} - \dots - a_{j_{m-1}}} \binom{n_{j_m} + a_{j_m}}{a_{j_m}} \left(\frac{p(j_m \rightarrow i)}{p(j_m \rightarrow \geq i)} \right)^{a_{j_m}} \left(\frac{p(j_m \rightarrow \geq i+1)}{p(j_m \rightarrow \geq i)} \right)^{n_{j_m}} p_k(n_1, \dots, n_{j_1-1}, n_{j_1} + a_{j_1}, n_{j_1+1}, \dots, n_{i-1})$$

This both looks horrible and is horribly inefficient. However, one can show that computing this probability is equivalent to computing the number of perfect matchings in a bipartite graph (for suitable choices of alternative splicing model and k). This again is believed to be impossible to do efficiently.

References

- [1] Q. Zhou, H. Chipperfield, D. A. Melton, and W. H. Wong. A gene regulatory network in mouse embryonic stem cells. *Proceedings of the National Academy of Sciences of the United States of America*, 104(42):16438–16443, 2007.