Algorithm

Main:

1. Choose a new edge i -> j not already in the graph.
    a. For edge competition, need to use “check for path” and “calculate IN(i)/OUT(j)” subroutines (see below).
2. Add the edge.
3. Check if larger SCC has formed.
    a. If yes, recompute GOUT/GIN/GSCC using BFS.
    b. Otherwise, update GOUT/GIN/GSCC (see below).
4. Return to 1 and repeat.

Check for path:

Beginning with nodes i,j, check if a path already exists from i to j. Use the following table. Blank in right-most column indicates that a path cannot exist.

	i in GIN	i in GOUT	j in GIN	j in GOUT	
0					Check for path i --> j using BFS.
1				T	Check for path i --> j using BFS.
2			T		
3			T	T	
4		T			
5		T		T	Check for path i --> j using BFS.
6		T	T		
7		T	T	T	
8	T				Using reversed edges, check for path j --> i using BFS.
9	T			T	Path must exists.
10	T		T		Using reversed edges, check for path j --> i using BFS.
11	T		T	T	Path must exists.
12	T	T			
13	T	T		T	Path must exists.
14	T	T	T		
15	T	T	T	T	Path must exists.

We choose the direction of the BFS in order to avoid the giant components (which would add a significant number of nodes to explore without adding any new information).

Calculate IN(i)/OUT(j):
Use the following tables:

|IN(i)|

i in GIN	i in GOUT	
		Perform full BFS starting from i using reversed edges.
	T	|IN(i)| = |GIN| + |IN(i) \ GIN| Find |IN(i) \ GIN| by doing BFS along reverse edges with all nodes in GIN marked as already visited.
T		Perform full BFS starting from i using reversed edges.
T	T	|IN(i)| = |GIN|

|OUT(j)|

j in GIN	j in GOUT	
		Perform full BFS starting from j.
	T	Perform full BFS starting from j.
T		|OUT(j)| = |GOUT| + |OUT(j) \ GOUT| Find |OUT(j) \ GOUT| by doing BFS with all nodes in GOUT marked as already visited.
T	T	|OUT(j)| = |GOUT|

Check for new SCC:
Given that edge i -> j was most recently added, find if a new SCC has formed and calculate its size. We use Tarjan's algorithm with either:

- Begin with j and do a forward BFS.
- Begin with i and do a reverse BFS (BFS using reversed edges).

Since Tarjan only does a single BFS, if |IN(i)| < |OUT(j)|, we would want to use the reversed graph to minimize the number of nodes visited. Without explicitly calculating these values we estimate their relative sizes using the membership relations to GIN/GOUT. Refer to the following table:

	i in GIN	i in GOUT	j in GIN	j in GOUT	
0					Forward BFS with OUT(j) (only if i not already path connected to j).

1				T	
2			T		Forward BFS with OUT(j).
3			T	T	
4		T			Backward BFS with IN(i).
5		T		T	Forward BFS with OUT(j) (only if i not already path connected to j).
6		T	T		
7		T	T	T	
8	T				
9	T			T	
10	T		T		Backward BFS with IN(i) (only if i not already path connected to j).
11	T		T	T	
12	T	T			
13	T	T		T	
14	T	T	T		
15	T	T	T	T	

All blank cases mean either a new SCC was not formed or the new SCC is a subset of the old GSCC.

Update GOUT/GIN/GSCC:

Once more, refer to the following table. Note that if we are adding nodes to GOUT, then we perform a BFS with all nodes in GOUT already marked as visited. Blank cases mean no nodes were added to GIN or GOUT.

	i in GIN	i in GOUT	j in GIN	j in GOUT	
0					
1				T	
2			T		Add IN(i) \ GIN to GIN.
3			T	T	Add IN(i) \ GIN to GIN.
4		T			Add OUT(j) \ GOUT to GOUT.
5		T		T	
6		T	T		Add OUT(j) \ GOUT to GOUT and any nodes already in GIN add to

					GSCC. Add IN(i) \ GIN to GIN and any nodes already in GOUT add to GSCC.
7		T	T	T	Add IN(i) \ GIN to GIN.
8	T				
9	T			T	
10	T		T		
11	T		T	T	
12	T	T			Add OUT(j) \ GOUT to GOUT.
13	T	T		T	
14	T	T	T		Add OUT(j) \ GOUT to GOUT.
15	T	T	T	T	

Comments:

- Runtime tests show algorithm scales as ~O(N^2.0) for structure-based selection and O(N^1.2) for random selection.
    - For 160k nodes, it takes about 30 minutes for structure-based selection and 12 seconds for random selection.