Appendix A: Network Generation Algorithms

The basic outline of network generation is described in Algorithm 1.

input : $N_F, PIG\_DAT, SF, P, F\_DIST, M\_MIX, K\_DAT, SHP\_DAT$
output: G_F, G_P

`// Assigns different operation types to individual farms uniformly at random.`

$F\_TYPE \leftarrow F\_TYPE\_GEN(F\_DIST, N_F)$;

`// Generates a directed farm graph with weighted (shipment rate) links.`

$G_F \leftarrow F\_GRAPH\_GEN(M\_MIX, F\_TYPE, N_F, K\_DAT, SHP\_DAT)$;

`// Assigns pigs to different operations based on data obtained from` $NASS$`. Scales the population by a factor` SF`.`

$PIG\_LIST \leftarrow PIG\_ALLOT(PIG\_DAT, N_F, SF)$;

`// Generates a pig level graph.`

$G_P \leftarrow P\_GRAPH\_GEN(G_F, N_F, PIG\_LIST, P)$;

Algorithm 1: Basic outline of graph generation

In the above box, N_F is the number of farms, PIG_DAT is the pig distribution data given in Table 4, SF is the scaling factor used to scale the pig level graph in order to make it computationally feasible in our model, P is the probability of within farm contacts between pigs, F_DIST is the farm type distribution shown in Table 3, M_MIX is the mixing matrix shown in Table 1, K_DAT contains the degree centrality data from Table 2, and SHP_DAT contains the mean and median shipment data. The farm level and pig level graphs are represented by G_F and G_P respectively. We describe the F_GRAPH_GEN function in more detail in Algorithm 2.

```
input  : M_MIX, F_TYPE, N_F, K_DAT, SHP_DAT
output : G_F
// Allocates in and out-degrees based on production types.
DEG_ALLOC ← DEG_GEN(F_TYPE, K_DAT);
for SRC_NODE ← 1 to N_F do
    // Get the allocated (max) out-degree of source node.
    MAX_KOUT = DEG_ALLOC[SRC_NODE, OUTDEG];
    for k ← 1 to MAX_KOUT do
        // Randomly sample the destination type from the mixing matrix
           distribution.
        DST_TYPE ∼ M_MIX[F_TYPE[SRC_NODE]];
        // Find all the nodes of the given type.
        DST_NODES ← {n : n ∈ [1, N_F] and F_TYPE(n) = DST_TYPE};
        // Pick the destination node which has the largest in-degree
           gap (max in-degree - current in-degree) to cover up.
        DST_NODE ← n ∈ DST_NODES that maximizes
         (KIN(n, MAX) − KIN(n, CUR));
        // generate shipment data from log-normal distribution with
           mean and median shipment values given in SHP_DAT.
        SHP_RATE ∼ Lognormal(SHP_DAT);
        // Create the link with generated shipment rate as weight.
        G_F ← G_F + EDGE(SRC_NODE, DST_NODE, SHP_RATE);
    end
end
```

Algorithm 2: F_GRAPH_GEN

We also describe another key component of the graph generator, P_GRAPH_GEN

in Algorithm 3, which generates the pig level network.

```
input  : G_F, N_F, PIG_LIST, P
output : G_P
for f ← 1 to N_F do
    // List of pigs in the farm f.
    PIG_NODES ← PIG_LIST(f);
    // Generate Erdos-Renyi graph for list of pig nodes with edge
       probability P.
    EDGES = ERDOS_RENYI(PIG_NODES, P);
    // Add the generated edges to the pig level graph.
    G_P ← G_P + EDGES;
end
// Normalize the edge weights to be used as probabilities.
G_F^NORM ← G_F / MAX(G_F);
for each (F1, F2) pair where F1, F2 ∈ G_F and F1 != F2 do
    // Probability of movement from farm F1 to any other farms.
    P_FROM ← G_F^NORM(F1, ALL);
    // Probability of movement from any farm to farm F2.
    P_TO ← G_F^NORM(ALL, F2);
    for each P1 ∈ PIG_LIST(F1) do
        for each P2 ∈ PIG_LIST(F2) do
            r ~ Uniform(0, 1);
            if r ≤ P_FROM × P_TO then
                G_P ← G_P + EDGE(P1, P2);
        end
    end
end
```

Algorithm 3: *P_GRAPH_GEN*