Table S1 BPA Reconstruction

Input: KEGG pathway with node list N and edge list E;

seeds = {};
Initialization cyclic graph G with nodes N and edges E;
Run Tarjan's algorithm to detect all cyclegroups (C_1,C_2,$\cdots$,C_n);
Remove all edges connecting nodes in the same cyclegroup;
for C_i in (C_1,C_2,$\cdots$,C_n) **do**
    Randomly number each node inside C_i from small to large;
    Add an edge from each lower number node to each higher number node;
    For each parent A of a member of the cycle group that is not itself in the cyclegroup, add an edge from A to each member of the cyclegroup;
end for

Output: Reconstructed pathway directed acyclic graph.

Table S2 PROPS Reconstruction

Input: KEGG pathway with node list N and edge list E;

seeds = {};
Initialization empty graph G with nodes N from the input pathway;
while E/= ∅ **do**
    Randomly select e_i from E;
    Add e_i to the graph G;
    if cyclic(G) **then**
        Remove e_i from the graph G;
    end if
    Remove e_i from the edge list E;
end while

Output: Reconstructed pathway directed acyclic graph.

Table S3 Clipper Reconstruction

Input: KEGG pathway with node list N and edge list E, healthy RNA-seq data H;
for n_i in N **do**
    Extract parent node set N_p for n_i according to the edge list;
        Extract corresponding expression data from healthy samples (H_{Np} and H_{ni}) for parent genes N_p and children gene n_i;

Fit linear regression model for healthy samples lm($H_{ni} \sim H_{Np}$);
for n_j in N_p **do**
Assign p-value p_{nj} to the edge from n_j to n_i;
end for **end for**
Sort the list of edges in ascending p-value order;
Initialization empty graph G with nodes N from the input pathway;
while E/= ∅ **do**
Add edge e_i to the graph G in ascending p-value order;
if cyclic(G) **then**
Remove e_i from the graph G;
end if
Remove e_i from the edge list E;
end while
Output: Reconstructed pathway directed acyclic graph.

Table S4 BNrich Reconstruction

Input: KEGG pathway with node list N and edge list E, healthy and disease RNA-seq data H and D;
Stage 1: Eliminate edges through biological approaches;
for e_i in E **do**
if (
e_i connects two nodes with same label;
e_i originates from nucleus resulting in cycle formation;
e_i is opposite of the signaling flux direction and would lead to cycle ;
) {
Remove e_i from the edge list;
}
end for
Stage 2: Simplify structure by LASSO
for n_i in N **do**
Extract parent node set N_p for n_i according to the edge list;
Extract corresponding expression data from healthy (H_{Np} and H_{ni}) and disease (D_{Np} and D_{ni}) samples for parent genes N_p and children gene n_i;
Fit regression model for healthy and disease sample separately LASSO($H_{ni} \sim H_{Np}$) and LASSO($D_{ni} \sim D_{Np}$);

for n_j in N_p **do**
  if $\beta_{nj} = 0$ in both models **then**
    Remove the edge from n_j to n_i;
  end if
end for
end for

Output: Simplified pathway directed acyclic graph.

Table S5 Ensemble Reconstruction

Input: KEGG pathway with node list N and edge list E;

seeds = {};
Initialization cyclic graph G with nodes N and edges E;
Run Tarjan's algorithm to detect all strongly connected components (SCCs) (SCC_1,SCC_2,⋯ ,SCC_n);
for metric in (TrueSkill, Social Agony) **do**
  for strategy in (Forward, Backward, Greedy) **do**
    for SCC_i in (SCC_1,SCC_2,⋯ ,SCC_n) **do**
      if metric == TrueSkill **then**
        For each edge (u,v):
          if v has a higher skill level than u (expected) **then**
            Update the skill level μ and σ of u by a small amount;
          else
            Update the skill level μ and σ of u to a large extent;
          end if
          The ranking score of node u is defined as $\mu_u - 3\sigma_u$;
        else if metric == Social Agony **then**
          Update ranking scores of each node by minimizing the total agony:
            $A(G) = min_r(\sum_{(u,v)\in E} \max(r(u) - r(v) + 1, 0))$
        end if
        if strategy == Forward **then**
          Select the node v which has the highest ranking score in SCC_i;
          Remove its all out edges in SCC_i;
        else if strategy == Backward **then**
          Select the node v which has the lowest ranking score in SCC_i;
          Remove its all in edges in SCC_i;
        else if strategy == Greedy **then**
          Select the edge with the highest difference in ranking score between the starting and ending nodes to remove;

28: **end if** 29: **end for** 30: **end for** 31: **end for**

32: For each edge (*u,v*), add the number of removals under the 6 conditions;

33: Remove edge in the order of highest to lowest voting scores until the graph becomes acyclic;

Output: Reconstructed pathway directed acyclic graph.