Report for Lab 2-Part 1

Name: Wu Yihang

StuID: A0285643W

_get_jt_clique_and_edges()

In this function, clique sets and their edges with each other are calculated. To be more detailed, `nx.Graph()` is used to represent the original graph. Then, we can get `jt_cliques` via `nx.find_cliques()`.

According to these cliques, we can build a clique graph, and here I add a helper function `build_clique_graph_and_weight(jt_cliques)`. The clique sets are needed as its input and `nx.Graph()` represents the clique graph, where cliques' indexes are nodes, and if 2 cliques have intersections, an edge between them will be added to the graph, with the size of their intersection being the edge's weight.

After getting the clique graph, `nx.maximum_spanning_tree()` is used to get the maximum spanning tree, then we can get `jt_edges` from this maximum spanning tree.

_get_clique_factors()

In this function, we get clique nodes' factors by using `factor_product()` and iteration.

_update_mrf_w_evidence()

In this function, `updated_factors` are obtained by using `factor_evidence()` and iteration. `query_nodes` are obtained by removing observed nodes from factor sets. `updated_edges` are obtained by removing edges related to observed nodes.

_get_clique_potentials()

In this function, the sum-product algorithm is used to get clique potentials. `nx.Graph()` is used to represent clique graph. Four helper functions: `collect(g,i,j,msg,cliques)`, `sendmessage(g,j,i,msg,cliques)`, `distribute(g,i,j,msg,cliques)` and `computeMarginal(g,i,msg)` are implemented like Lab 1.

_get_node_marginal_probabilities()

In this function, we get the marginal probability of each query node by marginal the clique potentials. **Considering a node may appear in multiple cliques of different sizes, we can first find the smallest clique related to it and then marginalize other nodes**, which can avoid extra computation.