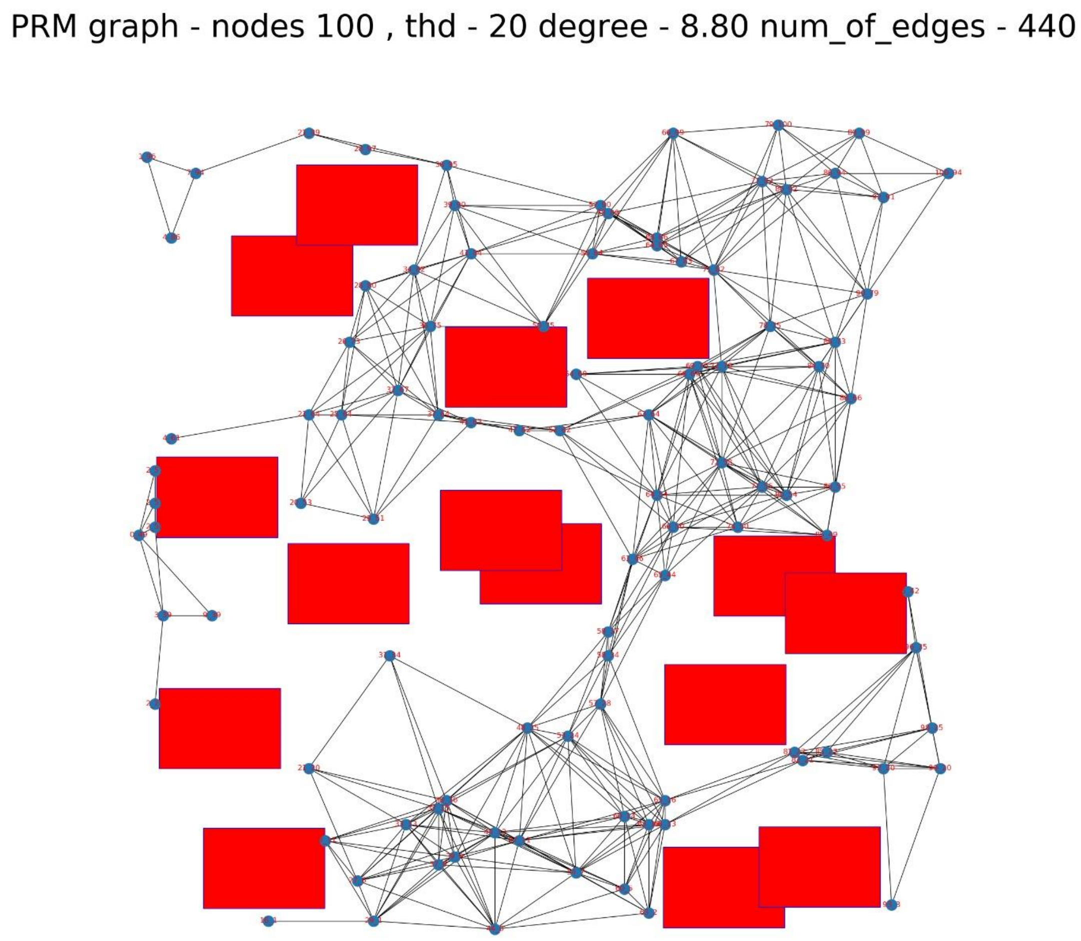
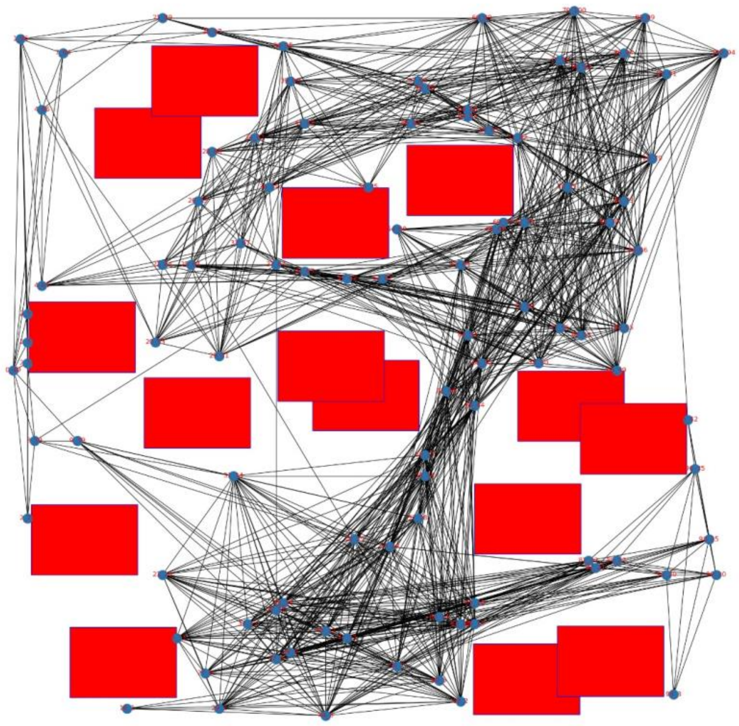
<u>Hands-on task</u>

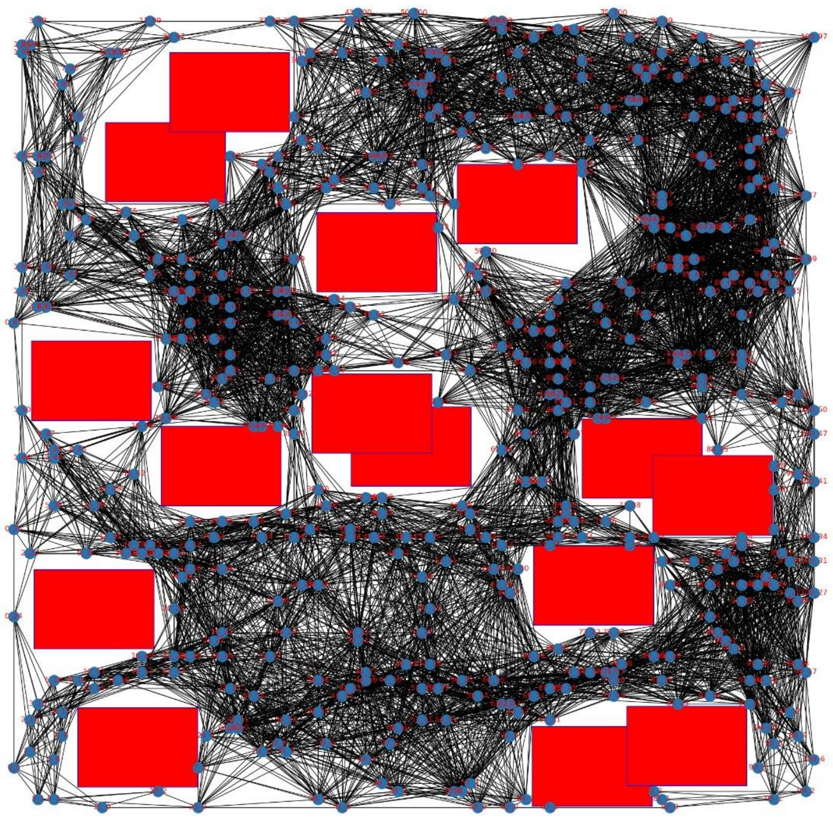
1. **PRM:**
   - **a.** The function GeneratePRM is implemented in PRM.py. (We chose to implement the algorithm such that the graph will hold N nodes, i.e. if a sampled node coordinates falls on an obstacle we would sample new coordinates until they are obstacle free).
   - **b.** The plots of different PRM's are provided below. Obstacles are in red, edges in black and nodes in blue. Number of edges and number of average node degree are indicated for each graph configuration.


PRM graph - nodes 100 , thd - 20 degree - 8.80 num_of_edges - 440

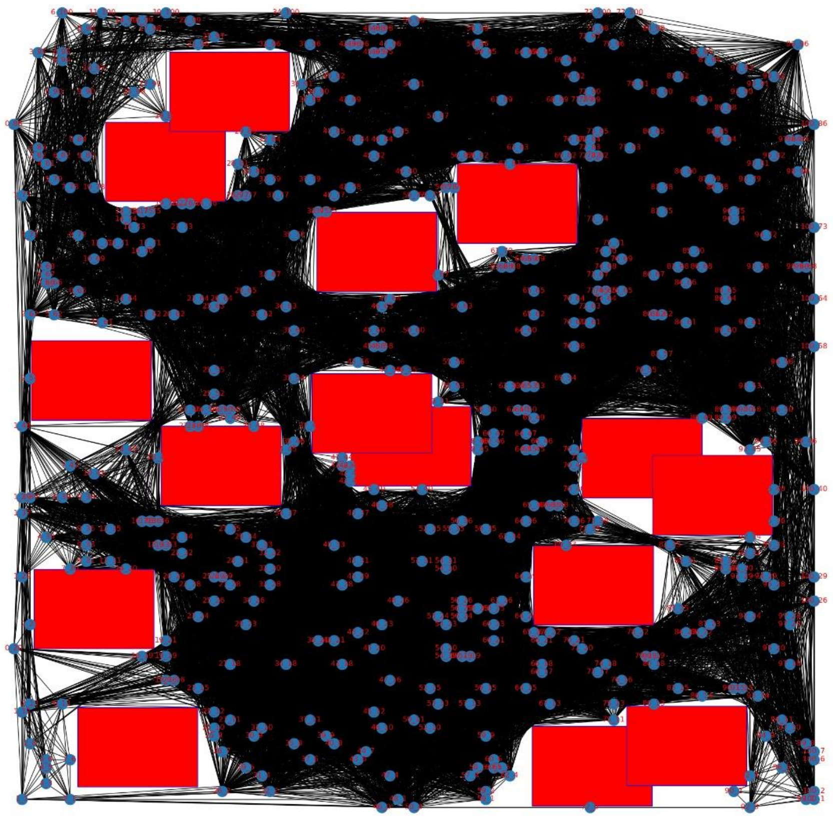
PRM graph - nodes 100 , thd - 50 degree - 23.96 num_of_edges - 1198



PRM graph - nodes 500 , thd - 20 degree - 42.99 num_of_edges - 10382



PRM graph - nodes 500 , thd - 50 degree - 112.53 num_of_edges - 27513



2. **Graph search:**
   We implemented the Dijkstra algorithm in GraphSearch.py.
   Edges are in yellow and shortest path edges are in green.
   The plot for the shortest path computed by Dijkstra:

PRM shortest_path graph - nodes 100 , thd 50 cost 130.02188426019978