Algorithm Description : (By Tze-Yuan Cheng)

In my algorithm, I implemented the roadmap approach of the sampling-based method, and then use Dijkstra Graph Search method to search for the path. .

(1)Sampling and Connecting Samples:

There are two stages in the sampling work. In the first stage, uniform distribution is used to get 1000 random collision free configuration sample, and then for each configuration, connect all the neighboring configurations within certain distance radius (in the code it is called "rho_dist"), and the "subdivision method" is applied to check the collision-free local path. By computing the number of neighboring configuration, the degree of connectivity of each sampled configuration can be computed, and the weight value for each sample is also determined (w(q)= 1/ 1+deg(q).) In the second stage of sampling, the configuration will be re-sampled with the probability = $\frac{w(q)}{\sum_{q' \in V} W(q')}$

To achieve this, I first accumulated weight of each configuration, and store the accumulated summation in a 1D array, in which the *i* th element of the array records the accumulated summation when $w(q_i)$ is added, and then a random number is generated from the range of (0 , $\sum_{q \in V} W(q)$). If the number falls on the interval of ($\sum_{i=1}^{k-1} w(q_i)$, $\sum_{i=1}^{k} w(q_i)$) in summation array, the *k* th configuration will be selected to generate local random sample around the *k* th configuration, and the local path will be reconnected using the subdivision method again.

(2)Checking Goal Connectivity via Graph Search

Once the configuration is connected with collision free edges, the Dijkstra graph search will be first used to find the shortest path leading to all the rewarding regions. My algorithm is ambitious to find the roadmap through which all the rewarding

regions can be reached, so if the searched roadmap can not connect all the rewarding regions to the initial robot position, the “solve” action will be restarted from sampling stage, until all the regions can be connected to robot. However, if the elapsed time exceeds the upper bound of running time, the “solve loop” will return a current roadmap solved.

(3)Determining the Final Path via Dijkstra Graph Search

Once solving action is complete, Dijkstra method will be implemented again to assemble the path segments. In the solving part, when the best path is found through Dijkstra, the order to “visit” each rewarding region will be recorded according to the distance from the initial position. Based on the order, the initial robot position will first be set as source to run Dijkstra method to find the first segment from initial to the nearest region; and then the second nearest region will be set as source to find second segment to third nearest region...In this fashion, the Dijkstra method will be run iteratively until all the path segments from *(n-1)* th region to *n* th region are found. Finally, all the path segments will be combined to be a final path after the “Get highest reward path” action is complete.