更改处

Graph_Node:

1. generate_g_value:

   因为任务目标为尽快到达终点,所以 g_value 应为与当前位置与速度有关的函数,

   此处将速度为正为负分开讨论:

```
dist_x = rand_end[0] - px
dist_y = rand_end[1] - py
if vx <= 0:
    g_x = dist_x - 2*vx + 1
else:
    g_x = dist_x / vx
if vy <= 0:
    g_y = dist_y - 2 * vy + 1
else:
    g_y = dist_y / vy
return max(g_x, g_y)
```

2. 0.1 概率为停留在原地， 0.9 的概率向前，修改函数 control

```
if not success: # 0.1 don't move
    vx = 0
    vy = 0

# dynamic model
else: #0.9 move
    vx = self.vx + ux
    vy = self.vy + uy
vx, vy = self.velocity_constraints(vx, vy)
px = self.px + vx
py = self.py + vy
```

dynamic_Programming:

1. track_the_best_plan
   1. 增加概率为 0.1 的随机选择节点
   2. 当添加起始点进入 trajectory list，重置 trajectory list

2. dynamic programming：
   1. 从终点节点向前依次更新路径上的节点
   2. g_value 更新值为：依赖下一节点的状态与当前更新节点的状态
   3.

```
plan = track_the_best_plan(track_map)
node_list = []
index = 0
for state in plan[ :: -1]:
  #Version1
  node_list.append(state)
  if state.is_goal:
    state.g_value = 0
  else:
    value_uk = []
    successor = node_list[index - 1]
    current_value = 0.9 * (1 + successor.g_value) + 0.1 * (1 + state.g_value)
    bellman_error += np.linalg.norm(state.g_value - current_value)
    state.g_value = current_value
  index += 1
```