**Abstract:**

This document presents an overview of two discrete planners which are used to search the goal. The planners considered are random planner and optimal planner. With specific conditions the planners are developed and also discussed further about respective complexity issues. Comprehensive coding is done in python.

**Introduction:**

Discrete motion planning is a well-established method and is known for computational complexity reduction.

Random planner and optimal planner gives the path from current robot position to target. Both the planners take the world state which is 2D grid representation and the clear path is represented as ‘0’ where the obstacle is defined as ‘1’. Along with the world state these planners take inputs like initial robot pose and goal pose.

Random planners try to make a move randomly to figure out a path for the goal pose. The random planner has few constraints like memory and also accept the solution in finite time (max\_step\_number). Considering only orthogonal moves the planner takes steps towards the goal pose.

Rather than going with random sequence of steps that leads to the goal pose, consider a solution that optimizes some criterion like time, distance, or memory. A cost function which indicates the cost accumulated for every step and also an action indicating to stop the search if not found are few things to be considered for optimal planning. Instead of random movement the planner takes the step based on this cost function. Here I’ve adapted A-star algorithm for optimal planner which includes calculation of heuristic function. Heuristic function always estimates the cost from current position to the goal.

**Discussion:**

The time complexity of optimal planner is O(n logn) while the time complexity of random planner is O(n) (by considering n as the maximum number of steps allowed). Random planner always tries to find the path quickly but fails to find the best path where as optimal planner though takes time in finding the path it always gives the best solution. There is always a tradeoff between these two.

Consider the target pose (5,5), to reach this goal pose the optimal planner gives the path as

[[2, 0], [2, 1], [3, 1], [4, 1], [4, 2], [4, 3], [5, 3], [5, 4], [5, 5]]

whereas random planner fails to reach the target because of its constraints like maximum step number.

While coming to test correctness of the code, the current code is said to be correct only when it acts according to the constraints. It needs to be correct at all conditions and for this I’ve checked its performance for different test cases. Conditions such as max\_step\_number or orthogonal moves are considered properly. All these things worked well at different cases.