Lab 5 Manual

# A\* and Hill Climbing

## 1. Introduction

In this lab, we will explore two fundamental search algorithms in Artificial Intelligence:

A\* Search

Hill Climbing (Faster, but may not find the best path)

Both algorithms are widely used in problem-solving, optimization, and pathfinding.

## 2. A\* Search

### 2.1 Overview

A\* (A-star) is an informed search algorithm that finds the optimal path from a start node to a goal node. It uses a combination of:

* **g(n)**: The actual cost from the start node to the current node n.
* **h(n)**: The estimated heuristic cost from n to the goal.
* **f(n) = g(n) + h(n)**: The total estimated cost of the path through n.

Initialize an open list (priority queue) and a closed list.

2. Add the start node to the open list with f(start) = g(start) + h(start).

3. While the open list is not empty:

a. Select the node with the lowest f(n) from the open list (current node).

b. If the current node is the goal, reconstruct the path and return it.

c. Move the current node to the closed list.

d. For each neighbor of the current node:

i. If the neighbor is in the closed list, skip it.

ii. Calculate tentative g(neighbor) = g(current) + cost(current, neighbor).

iii. If the neighbor is not in the open list or tentative g(neighbor) is lower:

- Update g(neighbor), h(neighbor), and f(neighbor).

- Set the current node as the parent of the neighbor.

- Add the neighbor to the open list.

4. If no path is found, return failure.

## 3 Hill Climbing Algorithm

### 3.1 Overview

Hill Climbing is a heuristic search algorithm that continuously moves towards the best possible state. It is an **optimization algorithm** that makes local changes to improve a given solution iteratively.

Types of Hill Climbing:

* **Simple Hill Climbing**: Evaluates one neighboring state at a time.
* **Steepest-Ascent Hill Climbing**: Evaluates all neighbors and picks the best one.
* **Stochastic Hill Climbing**: Selects a random neighbor with some probability.

1. Start with an initial solution (randomly generated state).

2. Loop until a solution is found or no improvements are possible:

a. Evaluate all possible neighbors.

b. Select the neighbor with the best improvement in the objective function.

c. If the new neighbor is better than the current state, move to it.

d. If no neighbor is better, return the current state as the best solution.