Evaluation of a Collection of Searches in Various Environments

# I. Performance, Environment, Actuators, and Sensors (PEAS) description and Environment Characteristics

## PEAS

**P**erformance Measure:

* The number of steps to the treasure and the number of steps back to the Entry/Exit point.
* Percentage of blocks traversed.
* How many are blocks used for the search.
* Did the Robot Agent succeed, take too long or get killed by a Rival Robot Agent.

**E**nvironment:

* Blocks that the Robot Agent can traverse.
* Blocks that contains obstacles that the Robot Agent cannot get over.
* One Treasure Block that contains the treasure for the Robot Agent to pick up.
* An Entry/Exit Point Block for the Robot Agent to use.
* Blocks where Rival Robot Agents reside.

**A**ctuators:

* Turn Left
* Turn Right
* Go Up
* Go Down
* Pick Up an Object

**S**ensors:

* Camera that tells Robot Agent what block it is currently in.

## Environment Characteristics

The environment is…

* Partially Observable – The Robot Agent can only tell where it is at.
* Deterministic – The Robot Agent can succeed or die in the environment depending on its actions.
* Sequential – The Robot Agent can only do one move at a time.
* Static – The environment does not change during the time the robot traverses it.
* Discrete – There is a finite number of states, time steps and actions.
* Single Agent – The Robot Agent is the only one operating the search. The other Rival Robot Agents can only stay in one block and do nothing unless the Robot Agent meets it in its block.
* Known – The Robot Agent does not much but it still knows the environment because it knows it cannot traverse obstacles, die if it meets a Rival Robot Agent and go home if it has the treasure.

# II. Problem Formulation for each Search Algorithm

The Problem Formulation defines the search problem with an initial state, a goal test, a successor function and a path cost for each search algorithm.

Random Search:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point and the treasure is somewhere in the environment.
* Goal Test – The Robot Agent is at the Entry/Exit Point with the treasure.
* Successor Function – Randomly pick a neighbor to travel to.
* Path Cost – Total number of time-steps

Depth-First Search [Tree Based]:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point and the treasure is somewhere in the environment.
* Goal Test – The Robot Agent is at the Entry/Exit Point with the treasure.
* Successor Function – Put Neighbors from your Left, Right, Up, Down direction into a LIFO queue and if goal test fails at the current block you are in, expand the deepest block in the LIFO queue.
* Path Cost – Total number of time-steps

Depth-First Search [Graph Based]:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point and the treasure is somewhere in the environment.
* Goal Test – The Robot Agent is at the Entry/Exit Point with the treasure.
* Successor Function – Put Neighbors from your Left, Right, Up, Down direction into a LIFO queue and if the current block you are in has already been traversed, go back to previous block. If not and the goal test fails, expand the deepest block in the LIFO queue.
* Path Cost – Total number of time-steps

Breadth-First Search [Tree Based]:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point and the treasure is somewhere in the environment.
* Goal Test – The Robot Agent is at the Entry/Exit Point with the treasure.
* Successor Function – Put Neighbors from your Left, Right, Up, Down direction into a FIFO queue and if goal test fails at the current block you are in, expand the shallowest block in the FIFO queue.
* Path Cost – Total number of time-steps

Breadth-First Search [Graph Based]:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point and the treasure is somewhere in the environment.
* Goal Test – Put Neighbors from your Left, Right, Up, Down direction into a FIFO queue and if the current block you are in has already been traversed, go back to previous block. If not and the goal test fails, expand the shallowest block in the FIFO queue.
* Path Cost – Total number of time-steps

Hill Climbing Search:

* Current State – All blocks in the environment have a path cost to the Treasure block/Entry-Exit block.
* Successor Function – If moving to Treasure block or to Entry-Exit block after picking up the treasure, check all neighbors and current block, select the lowest path cost and move to that block.
* Objective: Current Block’s path cost is zero, which means we have reached the Treasure block or Entry-Exit block.
* Path Cost on each Block: Straight line distance to Treasure Block or Entry/Exit Block.

Random Restart Hill Climbing Search:

* Current State – All blocks in the environment have a path cost to the Treasure block/Entry-Exit block.
* Successor Function – If moving to Treasure block or to Entry-Exit block after picking up the treasure, check all neighbors and current block, select the lowest path cost and move to that block. If lowest path cost is equal to current block’s path cost and current block’s path cost is not zero, restart the search at a different entry-exit point.
* Objective: Current Block’s path cost is zero, which means we have reached the Treasure block or Entry-Exit block.
* Path Cost on each Block: Straight line distance to Treasure Block or Entry/Exit Block.

Iterative Deepening Search:

* Initial State – An unexplored environment with the Robot Agent at the Entry/Exit Point, each block knowing their depth level and the treasure is somewhere in the environment.
* Goal Test – The Robot Agent is at the Entry/Exit Point with the treasure.
* Successor Function – Build Tree to a limit. Search the tree for treasure. If the treasure is not in the current tree, increase limit by one and rebuild the tree. Once the treasure is found, search for the home block from the treasure block. Repeat building tree with limit until the home block is found.
* Path Cost – Total number of time-steps

A\* Search:

* Current State – All blocks in the environment have a path cost to the Treasure block/Entry-Exit block.
* Successor Function – Collect the neighbors of the current block. Compare the path cost of each block and select the block with the least expensive path cost. Increase the cost to get to that block by one. This ensures that we do not revisit that block in a future successor function call.
* Objective: The current block’s path cost is zero, which means we have found the shortest path to the treasure block/home block.
* Path Cost on each Block: Straight line distance to Treasure Block or Entry/Exit Block.

Min-Conflict:

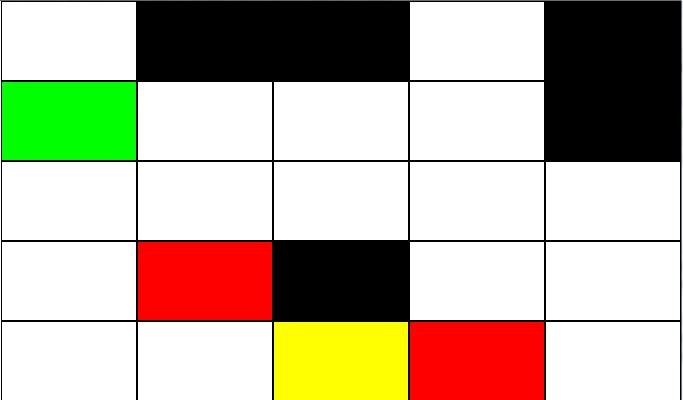
* Current State – All blocks are empty, an agent, an obstacle, a starting point, or a block that contains the Treasure.
* Successor Function – Collect the neighbors of the current block. The constraint is to not collect neighbors that are an obstacle, an agent or has been visited before during both searches (the search to Goal, and the search to Home).
* Objective: The Robot Agent is at the Entry/Exit Block with the Treasure in hand.

## III. Performance Measure

To measure how much memory is used for each search, a block in the environment is estimated to be around 44 bytes. The environment may have boundaries but since the Robot Agent only knows the block it is currently at, the environment can be considered an undirected graph where infinite loops can happen. The space and time requirements will use complexities of the actual run and of the worst-case scenarios for each search algorithm that performs on a specific environment size. For the actual runs, the space complexity is considered as the maximum amount of blocks the search had in the queue/look-ahead at one given time and the time complexity is considered as total number of blocks traversed during run-time.

For the first five graphs, the environment consists of obstacles which take up 20% of the graph and Rival Robot Agents that take up 10% of the graph.

### 5x5 Environment



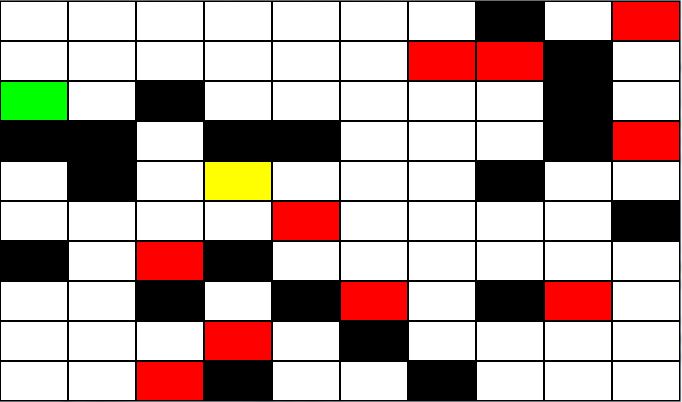
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­5), Space: O(45)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(45), Space: O(4\*5)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(45), Space: O(4\*5)  
A\* – Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (5x5) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative Deepening | A\* | Min  Conflict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Number of Allowed Steps | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 |
| Number of Steps The Robot Took Before Object Pick-Up | 24 | 31 | 23 | 1250 | 14 | 3 | 3 | 23 | 3 | 5 |
| Number of Steps The Robot Took After Object Pick-Up | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 |
| Total Steps Taken | 31 | 31 | 23 | 1250 | 14 | 3 | 3 | 23 | 3 | 14 |
| % of Blocks Traversed | 19.00 | 32.00 | 28.00 | 8.00 | 36.00 | 1.00 | 8.00 | 32.00 | 12.00 | 40.00 |
| Did Robot Die? | YES | YES | YES | NO | YES | NO | YES | YES | YES | NO |
| Did Robot Succeed? | NO | NO | NO | NO | NO | NO | NO | NO | NO | YES |
| Number of Blocks Used (TIME) | 31 | 16 | 11 | 1251 | 14 | 4 | 4 | 23 | 3 | 14 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 33 | 12 | 3125 | 11 | 5 | 5 | 18 | 6 | 4 |
| Total Memory Used (Bytes) | 1364 | 704 | 484 | 55044 | 616 | 176 | 176 | 1012 | 132 | 616 |

### 10x10 Environment



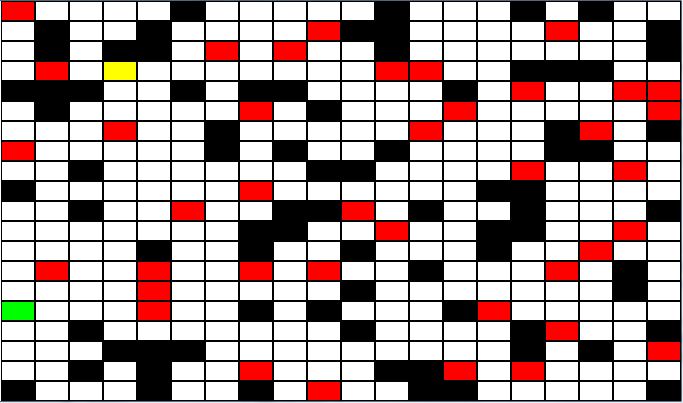
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­10), Space: O(410)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(410), Space: O(4\*10)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(410), Space: O(4\*10)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (10x10) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative Deepening | A \* | Min Con flict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Number of Allowed Steps | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 |
| Number of Steps The Robot Took Before Object Pick-Up | 59 | 5000 | 124 | 5000 | 13 | 5000 | 16 | 102 | 8 | 12 |
| Number of Steps The Robot Took After Object Pick-Up | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total Steps Taken | 59 | 5000 | 124 | 5000 | 13 | 5000 | 16 | 102 | 8 | 12 |
| % of Blocks Traversed | 19.00 | 15.00 | 16.00 | 2.00 | 10.00 | 1.00 | 8.00 | 20.00 | 8.0 | 11.00 |
| Did Robot Die? | YES | NO | YES | NO | YES | NO | YES | YES | YES | NO |
| Did Robot Succeed? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Number of Blocks Used (TIME) | 31 | 2571 | 64 | 5001 | 14 | 4 | 4 | 102 | 8 | 12 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 5946 | 1980 | 12500 | 10 | 5 | 5 | 54 | 19 | 4 |
| Total Memory Used (Bytes) | 1364 | 113124 | 2816 | 220044 | 616 | 176 | 176 | 4488 | 352 | 528 |

### 20x20 Environment



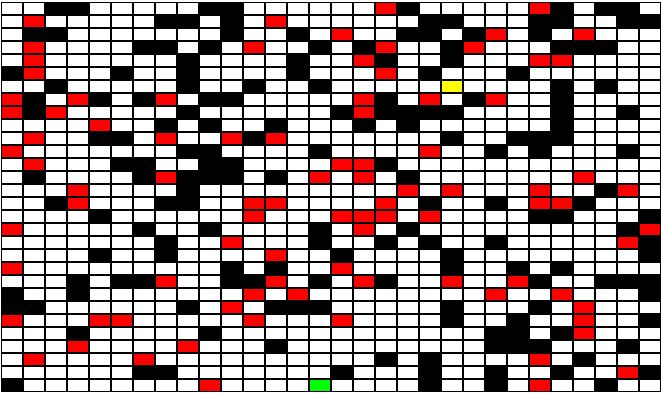
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­20), Space: O(420)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(420), Space: O(4\*20)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(420), Space: O(4\*20)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (20x20) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min Conf lict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Number of Allowed Steps | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 |
| Number of Steps The Robot Took Before Object Pick-Up | 92 | 57 | 35 | 20000 | 4 | 20000 | 25 | 30 | 9 | 25 |
| Number of Steps The Robot Took After Object Pick-Up | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total Steps Taken | 92 | 57 | 35 | 20000 | 4 | 20000 | 25 | 30 | 9 | 25 |
| % of Blocks Traversed | 19.00 | 2.75 | 2.75 | 0.50 | 0.75 | 1.00 | 8.00 | 3.0 | 1.75 | 6.00 |
| Did Robot Die? | YES | YES | YES | NO | YES | NO | YES | YES | YES | NO |
| Did Robot Succeed? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Number of Blocks Used (TIME) | 31 | 30 | 14 | 20001 | 5 | 4 | 4 | 30 | 9 | 25 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 70 | 19 | 50000 | 8 | 5 | 5 | 20 | 16 | 4 |
| Total Memory Used (Bytes) | 1364 | 1320 | 616 | 880044 | 220 | 176 | 176 | 1320 | 396 | 1100 |

### 30x30 Environment



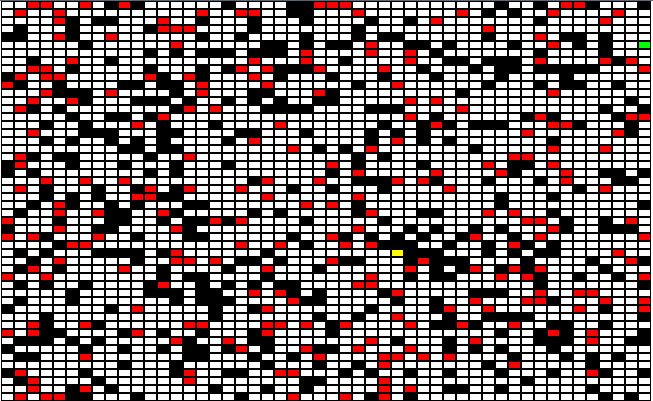
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­30), Space: O(430)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(430), Space: O(4\*30)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(430), Space: O(4\*30)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (30x30) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min Conf lict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Number of Allowed Steps | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 |
| Number of Steps The Robot Took Before Object Pick-Up | 119 | 1339 | 197 | 5 | 5 | 45000 | 7 | 112 | 7 | 29 |
| Number of Steps The Robot Took After Object Pick-Up | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total Steps Taken | 119 | 1339 | 197 | 5 | 5 | 45000 | 7 | 112 | 7 | 29 |
| % of Blocks Traversed | 19.00 | 3.44 | 3.33 | 0.44 | 0.44 | 1.00 | 8.00 | 3.44 | 0.55 | 3.11 |
| Did Robot Die? | YES | YES | YES | YES | YES | NO | YES | YES | YES | NO |
| Did Robot Succeed? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Number of Blocks Used (TIME) | 31 | 644 | 62 | 6 | 6 | 4 | 4 | 112 | 7 | 29 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 1752 | 141 | 10 | 6 | 5 | 5 | 47 | 14 | 4 |
| Total Memory Used (Bytes) | 1364 | 28336 | 2728 | 264 | 264 | 176 | 176 | 4928 | 308 | 1276 |

### 50x50 Environment



#### Space and Time Requirements

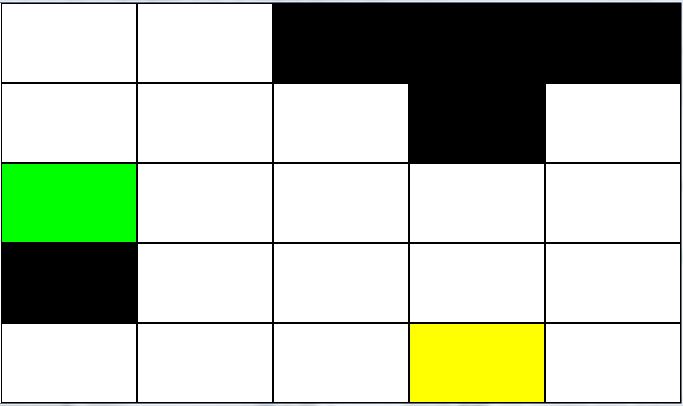
**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­50), Space: O(450)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(450), Space: O(4\*50)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(450), Space: O(4\*50)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (50x50) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min Con flict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Number of Allowed Steps | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | - |
| Number of Steps The Robot Took Before Object Pick-Up | 13 | 31 | 23 | 125000 | 20 | 3 | 3 | 20 | 3 | 9 |
| Number of Steps The Robot Took After Object Pick-Up | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total Steps Taken | 13 | 31 | 23 | 125000 | 20 | 3 | 3 | 20 | 3 | 9 |
| % of Blocks Traversed | 19.00 | 0.32 | 0.32 | 0.12 | 0.67 | 1.00 | 8.00 | 0.36 | 0.12 | 0.32 |
| Did Robot Die? | YES | YES | YES | NO | YES | NO | YES | YES | YES | NO |
| Did Robot Succeed? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Number of Blocks Used (TIME) | 31 | 15 | 10 | 125001 | 21 | 4 | 4 | 20 | 3 | 9 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 34 | 14 | 208335 | 28 | 5 | 5 | 21 | 6 | 4 |
| Total Memory Used (Bytes) | 1364 | 660 | 440 | 5500044 | 924 | 176 | 176 | 880 | 132 | 396 |

For the last five environments, the environments will vary in size and only have 20% of its space filled with obstacles and no agents involved. This will allow us to compare the performances of the search algorithms with no chance of dying, but they can get stuck in between obstacles.

## 5x5 Environment, No Agents



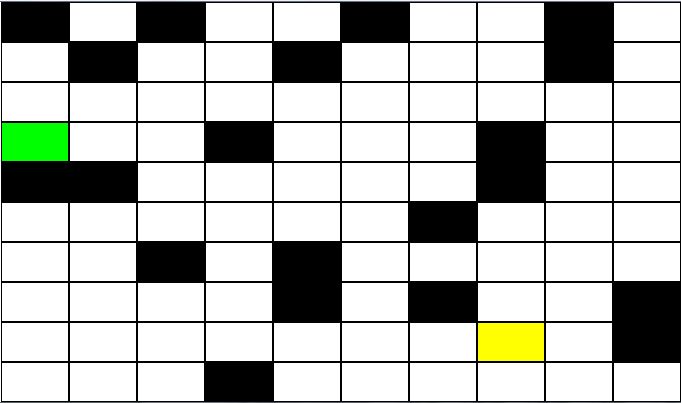
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­5), Space: O(45)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(45), Space: O(4\*5)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(45), Space: O(4\*5)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (5x5) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min Con flict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Number of Allowed Steps | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 | 1250 |
| Number of Steps The Robot Took Before Object Pick-Up | 86 | 860 | 120 | 1250 | 19 | 1250 | 15 | 72 | 6 | 6 |
| Number of Steps The Robot Took After Object Pick-Up | 194 | 391 | 76 | 0 | 15 | 0 | 2 | 75 | 5 | 0 |
| Total Steps Taken | 280 | 1251 | 196 | 1250 | 34 | 1250 | 17 | 147 | 11 | 6 |
| % of Blocks Traversed | 80.00 | 80.00 | 80.00 | 8.00 | 56.00 | 4.00 | 28.00 | 80.00 | 28.00 | 20.00 |
| Did Robot Die? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Did Robot Succeed? | YES | NO | YES | NO | YES | NO | YES | YES | YES | NO |
| Number of Blocks Used (TIME) | 280 | 621 | 88 | 1251 | 33 | 1 | 18 | 147 | 11 | 6 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 1105 | 220 | 3125 | 12 | 5 | 5 | 27 | 11 | 4 |
| Total Memory Used (Bytes) | 12320 | 27324 | 3872 | 55044 | 1452 | 44 | 792 | 6468 | 484 | 264 |

## 10x10, No Agents



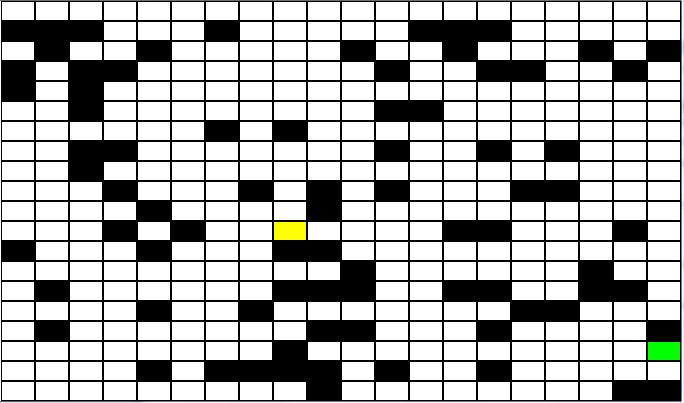
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­10), Space: O(410)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(410), Space: O(4\*10)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(410), Space: O(4\*10)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (10x10) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min Conflict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| # of Allowed Steps | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 |
| Number of Steps The Robot Took Before Object Pick-Up | 258 | 5000 | 5000 | 5000 | 61 | 5000 | 172 | 592 | 12 | 3 |
| Number of Steps The Robot Took After Object Pick-Up | 550 | 0 | 0 | 0 | 49 | 0 | 1 | 661 | 12 | 0 |
| Total Steps Taken | 780 | 5000 | 5000 | 5000 | 110 | 5000 | 173 | 1253 | 24 | 3 |
| % of Blocks Traversed | 60.00 | 25.00 | 41.00 | 2.00 | 55.00 | 1.00 | 36.00 | 79.00 | 24.00 | .03 |
| Did Robot Die? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Did Robot Succeed? | YES | NO | NO | NO | YES | NO | YES | YES | YES | NO |
| Number of Blocks Used (TIME) | 780 | 2603 | 794 | 5001 | 103 | 1 | 174 | 1253 | 12 | 3 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 7048 | 11602 | 12500 | 52 | 5 | 5 | 112 | 42 | 4 |
| Total Memory Used (Bytes) | 34320 | 114532 | 34936 | 220044 | 4532 | 44 | 7656 | 55132 | 528 | 176 |

## 20x20, No Agents



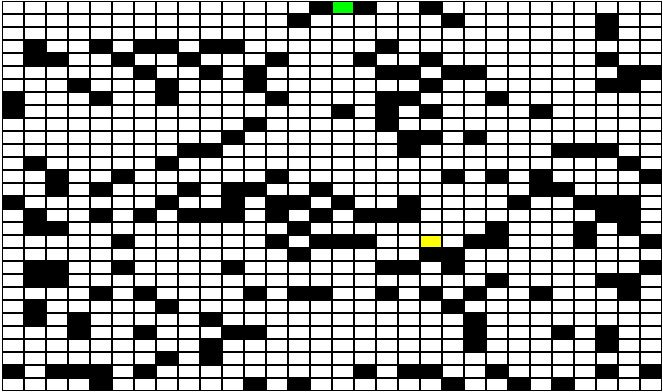
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­20), Space: O(420)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(420), Space: O(4\*20)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(420), Space: O(4\*20)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (20x20) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min  Conflict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| # of Allowed Steps | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 | 20000 |
| Number of Steps The Robot Took Before Object Pick-Up | 997 | 20000 | 20000 | 20000 | 116 | 20000 | 20000 | 1507 | 17 | 13 |
| Number of Steps The Robot Took After Object Pick-Up | 16558 | 0 | 0 | 0 | 638 | 0 | 0 | 3587 | 17 | 0 |
| Total Steps Taken | 17585 | 20000 | 20000 | 20000 | 754 | 20000 | 20000 | 5094 | 34 | 13 |
| % of Blocks Traversed | 78.50 | 6.75 | 13.00 | 2.75 | 58.50 | 0.25 | 31.75 | 76.50 | 4.25 | 3.00 |
| Did Robot Die? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Did Robot Succeed? | YES | NO | NO | NO | YES | NO | NO | YES | YES | NO |
| Number of Blocks Used (TIME) | 17585 | 9304 | 9830 | 20002 | 527 | 1 | 20000 | 5094 | 34 | 13 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 26267 | 64251 | 33347 | 160 | 5 | 5 | 460 | 51 | 4 |
| Total Memory Used (Bytes) | 773740 | 409376 | 432520 | 880088 | 23188 | 44 | 880000 | 224136 | 1496 | 572 |

## 30x30, No Agents



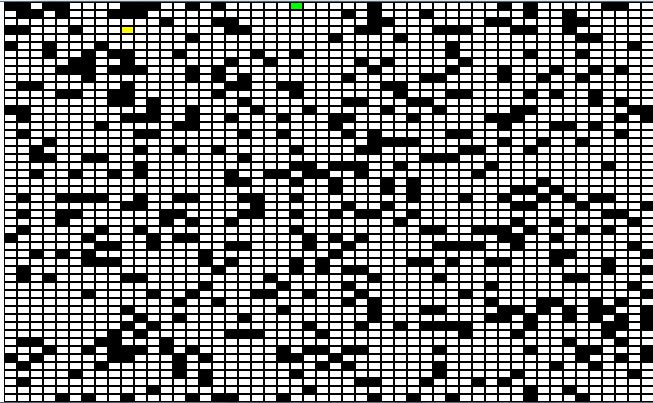
#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­30), Space: O(430)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(430), Space: O(4\*30)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(430), Space: O(4\*30)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (30x30) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | Min  Conf  Lict |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Number of Allowed Steps | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 | 45000 |
| Number of Steps The Robot Took Before Object Pick-Up | 278 | 45000 | 45000 | 45000 | 1533 | 45000 | 45000 | 4710 | 22 | 11 |
| Number of Steps The Robot Took After Object Pick-Up | 2927 | 0 | 0 | 0 | 859 | 0 | 0 | 7651 | 22 | 0 |
| Total Steps Taken | 3205 | 45000 | 45000 | 45000 | 2392 | 45000 | 45000 | 12361 | 44 | 11 |
| % of Blocks Traversed | 46.11 | 6.22 | 10.77 | 0.33 | 66.00 | 0.11 | 20.77 | 74.66 | 4.88 | 1.11 |
| Did Robot Die? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Did Robot Succeed? | YES | NO | NO | NO | YES | NO | NO | YES | YES | NO |
| Number of Blocks Used (TIME) | 3205 | 21057 | 20037 | 45001 | 1756 | 1 | 45000 | 12361 | 44 | 11 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 62108 | 223432 | 74996 | 490 | 5 | 5 | 875 | 66 | 4 |
| Total Memory Used (Bytes) | 141020 | 926508 | 881628 | 1980044 | 77264 | 44 | 1980000 | 543884 | 1936 | 484 |

## 50x50, No Agents



#### Space and Time Requirements

**Worst-case Scenario:**

Random Search – Time: O(n), Space: O(n)  
Breadth-First Search [Tree] – Time: O(inf.), Space: O(inf.)  
Breadth-First Search [Graph] – Time: O(4­50), Space: O(450)  
Depth-First Search [Tree] – Time: O(inf.), Space(inf.)  
Depth-First Search [Graph] – Time: O(450), Space: O(4\*50)  
Hill Climbing – Time: O(n), Space: O(5)  
Random Restart Hill Climbing – Time: O(n), Space: O(5)  
Iterative Deepening – Time: O(450), Space: O(4\*50)  
A\* -- Time: O(relative error \* length of optimal path), Space: Keeps all nodes in memory  
Min-Conflict – Time: O(n), Space: O(4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Size (50x50) | Random Search | BFS [Tree] | BFS [Graph] | DFS [Tree] | DFS [Graph] | Hill Climbing | Random Restart Hill Climbing | Iterative  Deepening | A\* | M  C. |
| Obstacle % | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Agent % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Number of Allowed Steps | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | 125000 | - |
| Number of Steps The Robot Took Before Object Pick-Up | 321 | 125000 | 125000 | 125000 | 97 | 125000 | 17 | 2276 | 16 | 48 |
| Number of Steps The Robot Took After Object Pick-Up | 1879 | 0 | 0 | 0 | 130 | 0 | 124948 | 2528 | 16 | 0 |
| Total Steps Taken | 2200 | 125000 | 125000 | 125000 | 227 | 125000 | 125000 | 4804 | 32 | 48 |
| % of Blocks Traversed | 12.64 | 2.84 | 5.48 | 0.24 | 3.32 | 0.24 | 0.91 | 15.47 | 1.28 | 1.88 |
| Did Robot Die? | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO |
| Did Robot Succeed? | YES | NO | NO | NO | YES | NO | NO | YES | YES | NO |
| Number of Blocks Used (TIME) | 2200 | 55837 | 61490 | 125001 | 202 | 6 | 24 | 4804 | 16 | 48 |
| Max Number of Blocks in Memory during Run-Time (SPACE) | 1 | 156902 | 292530 | 125005 | 80 | 5 | 5 | 388 | 48 | 4 |
| Total Memory Used (Bytes) | 96800 | 2456828 | 2705560 | 5500044 | 8888 | 264 | 1056 | 211376 | 704 | 2112 |

## III. Analysis

In comparison to the last report, the following analysis still holds true. When agents are involved in the environment, the searches are shown to fail no matter how well they perform. Despite that fact, when it comes to performance in comparison of space and time complexity, the local searches perform exceptionally well unless obstacles are covered around the block that contains the treasure. The local searches only use up to 5 blocks at a time for each movement decision so in comparison to breadth-first search or depth-first search, its performance stands out dramatically. The A\* search does not only use 5 blocks at a time but instead, it uses a queue to find the least expensive path.

When it comes to environments with no agents involved, the random search always picked up and returned the treasure if no time limit was involved. The same can be said for the depth-first and breadth-first search but the problem is that if no state-checking is involved for repeated blocks, the two searches perform terribly.

Random restart hill-climbing solved three out of the ten environments (**30%**). A\* solved ten out of ten environments (**100%**). Min-Conflict solved none of the environments. Hill-climbing solved none of them but that is another issue that should be addressed in this analysis of the search algorithms. The layout of the environment can hinder the performance of the algorithms, including most local searches. The heuristics of the local searches such as hill-climbing can make the searches fail considering that the robot should go around the obstacles but if an obstacle is in the way of the optimal path or if the robot gets stuck between three obstacles, the search can get stuck. The same thing can happen to the random restart search because you can have a case where the obstacles block all optimal paths to the treasure block. This case is actually examined in one of the non-agent environments above.

Another problem that was observed in these simulations is that if you are using a large environment, breadth-first search and depth-first search can explode in space and time complexities. Another problem that has occurred is that the local searches have a higher chance of getting stuck when the amount of obstacles increase.

|  |
| --- |
| **Search Rankings** |
| A\* |
| Random Restart Hill-Climbing |
| Hill-Climbing |
| Depth-First Search (Graph Version) |
| Breadth-First Search (Graph Version) |
| Iterative Deepening |
| Breadth-First Search (Tree Version) |
| Random Search |
| Min-Conflict |
| Depth-First Search (Tree Version) |

## IV. Conclusion

Overall, the best search algorithm really depends on what kind of environment you are dealing with. Do you see a clean path to the treasure block and back? Do local searches such as hill climbing or random restart hill climbing. Is the environment cluttered with obstacles to where there are large obstructions to the treasure block and back? You may need to do a random search, depth-first search or breadth-first search or an iterative deepening search. I recommend a state-checked depth-first search since the performance is linear and stable. Are agents are involved in the environment? You may be better off with a local search even though it will probably die anyways because you have a chance of an agent being on an optimal path on the way to the treasure block and on the way back to the entry/exit point.