FlatLand Assignment(Motion Planning Application) – Motion Planning (RBE550 Spring 2023)

Rohin Siddhartha Palaniappan Venkateswaran

Instructions to run code

Change the grid_size and coverage to any arbitrary value. If no arguments, default arguments are given as grid_size=128 and coverage=10%.

Note: The algorithms implemented in this assignment are capped at coverage of 40%. Any other coverage value given above 40% will result in error.

```
python3 -u flatland.py --grid_size 128 --coverage 5
```

Source code and Algorithm design and summary

The various functions that have been written and their description is as follows:

choose_start_and_goal: This function arbitrarily selects a start and goal location in the grid. The grid which is used in this assignment has been built in the previous "Turtles" assignment.

create_adjacency_list: This function creates a dictionary where the keys are the vertices in sequential order and each key's value corresponds to the vertices which can be connected to this vertex under consideration. Valid neighbours are checked in 4 directions and 8 directions, in 2 separate functions.

Once the adjacency list is created, it is passed to the **bfs,dfs,random_planner and djikstra functions** along with the start and goal locations from above. These functions calculate the path from start to goal locations. Djikstra Algorithm uses a weighted graph, and diagonal moves are allowed, hence a weighted adjacency list is passed to this algorithm.

The calculated path is returned from these functions in the main, and are plotted. Visualizations are shown in the section below.

Results

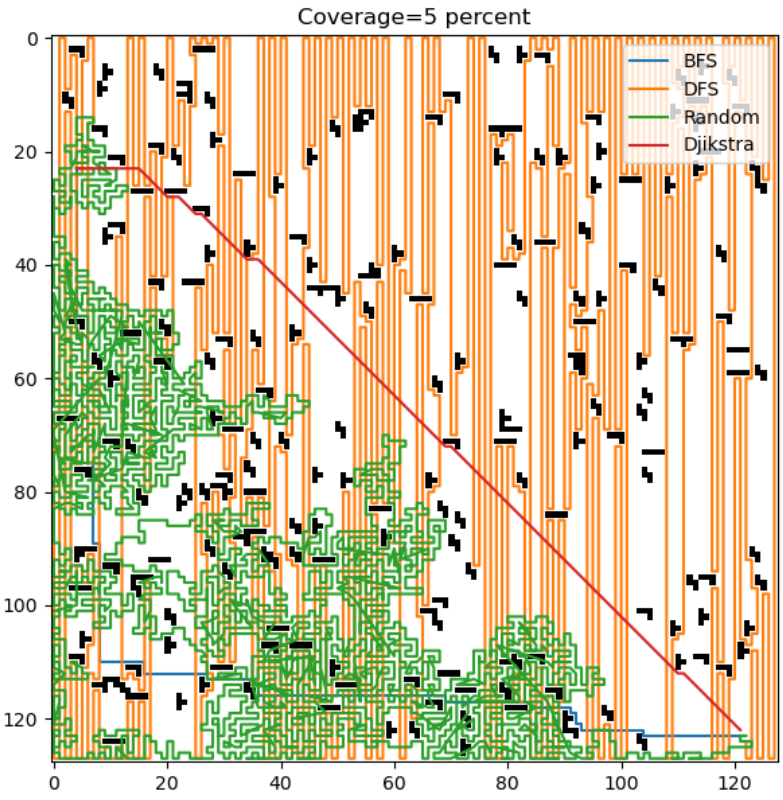

Figure 1: 5 Percent Coverage

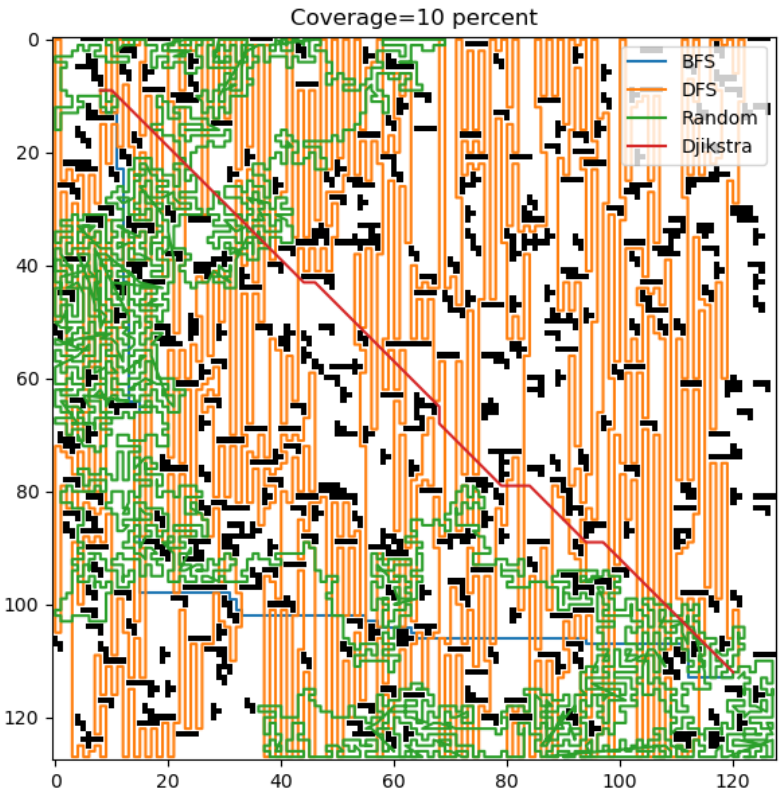

Figure 2: 10 Percent Coverage

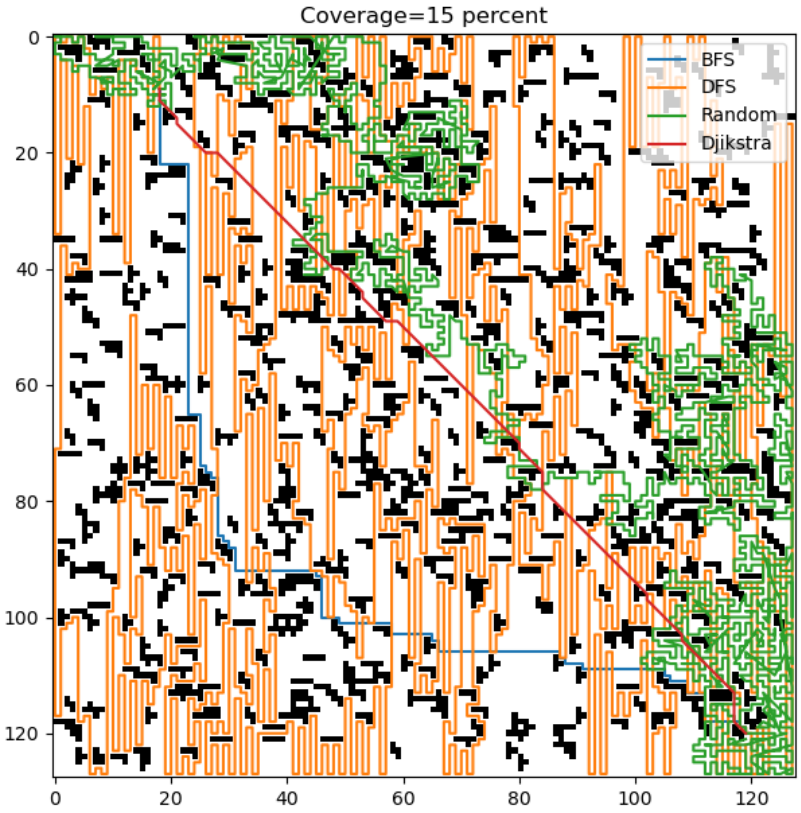

Figure 3: 15 Percent Coverage

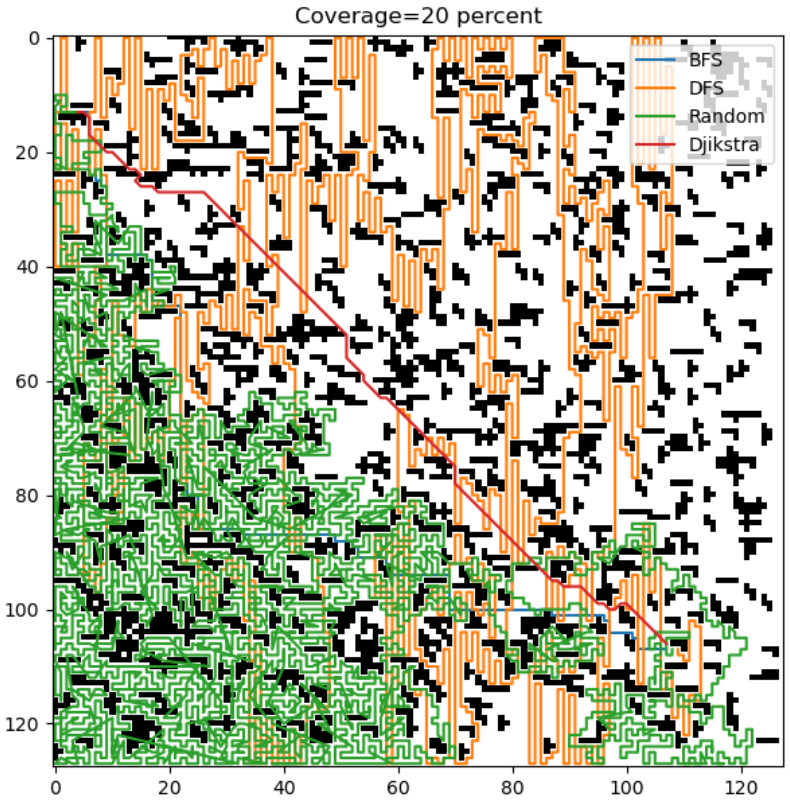

Figure 4: 20 Percent Coverage

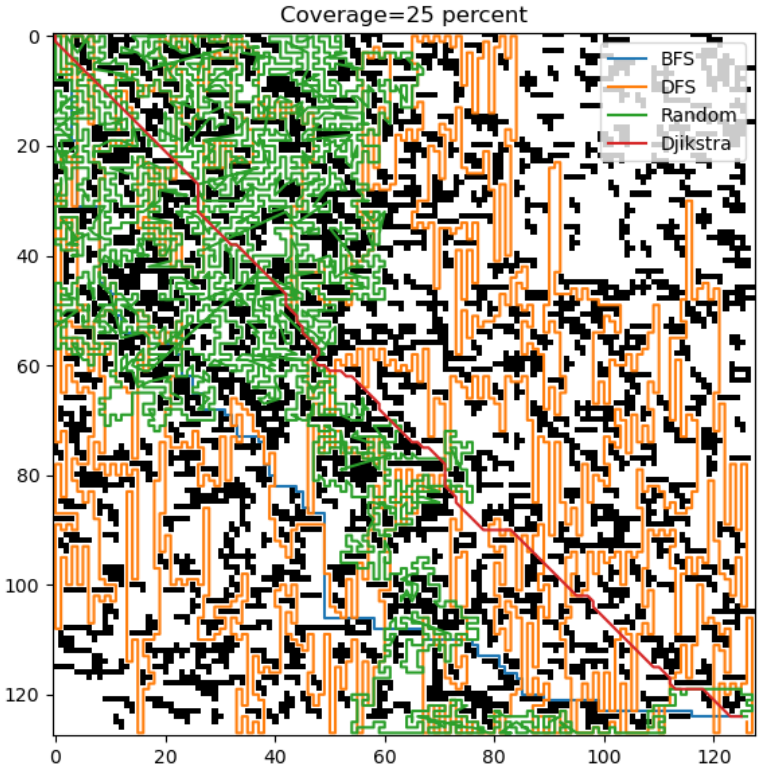

Figure 5: 25 Percent Coverage

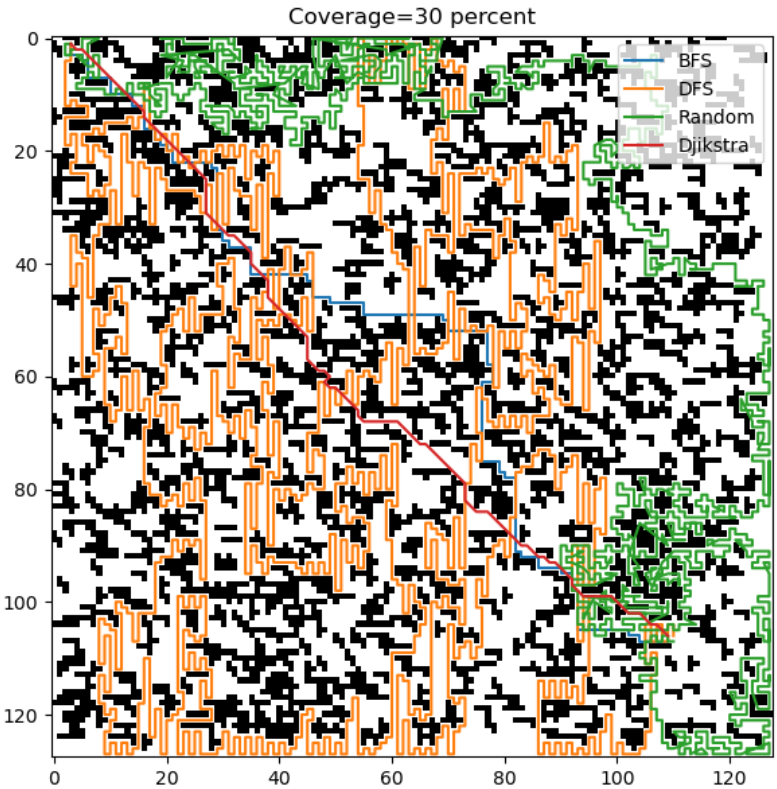

Figure 6: 30 Percent Coverage

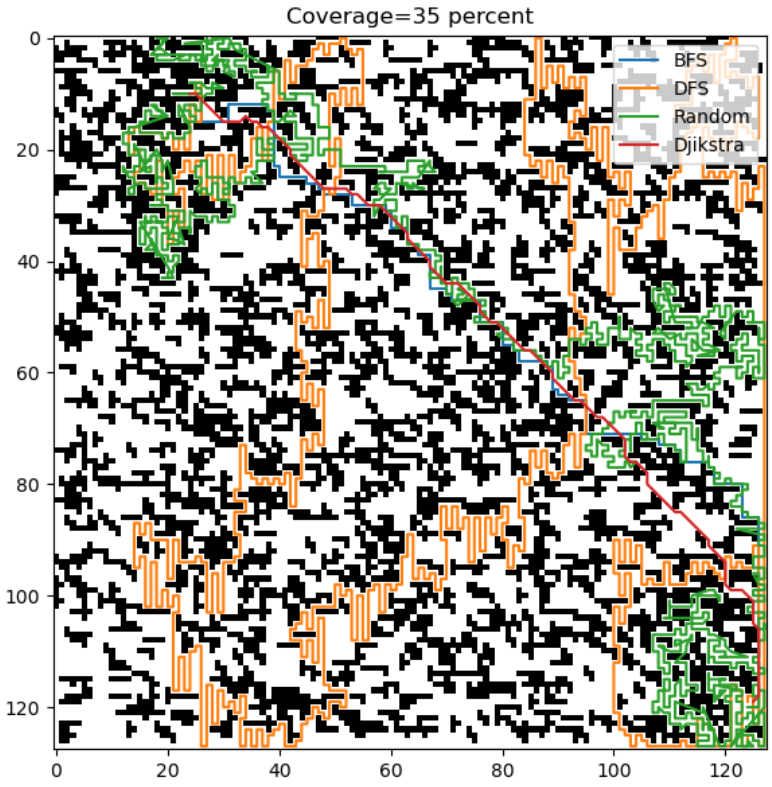

Figure 7: 35 Percent Coverage

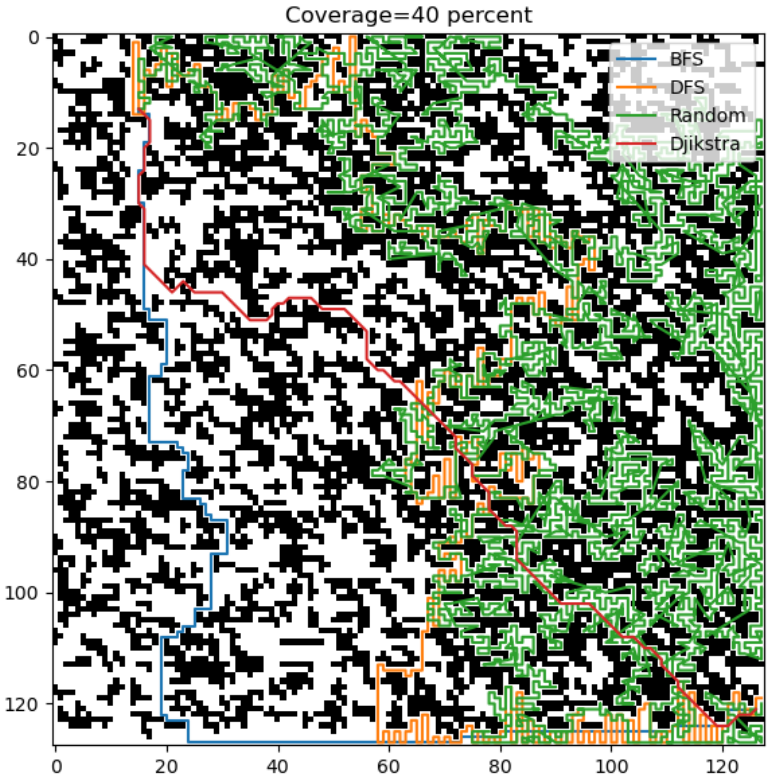

Figure 8: 40 Percent Coverage

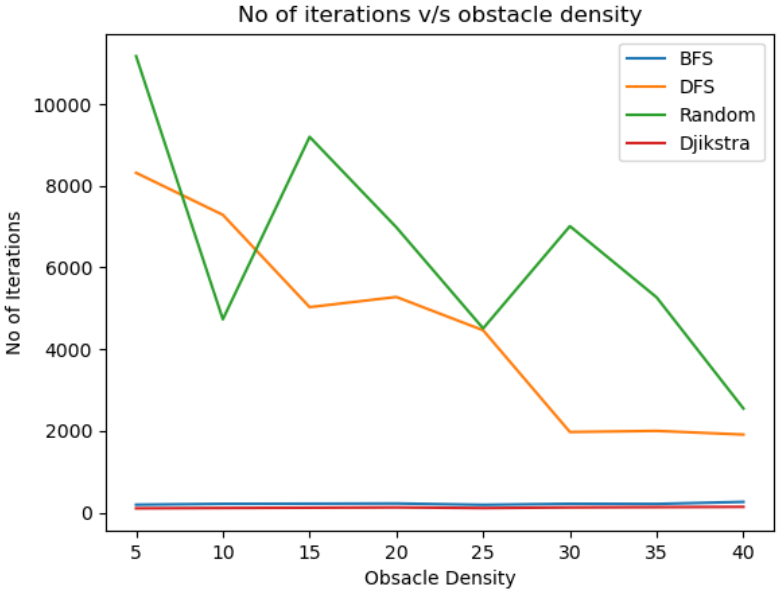

Figure 9: No of iterations v/s obstacle density