Visualizing Generic Graph Spaces

Tejus C[1], Yashashav DK[2], Ravi Kumar S[3]
[1, 2, 3] Dept of Information Science and Engineering
[1, 2, 3] Atria Institute of Technology

Abstract- *A generic grid based graph space visualizer is presented. The interface provides a visual link over the search space for any user-mentioned source and target pairs. Source and target vertices can be dragged across the grid space. Well defined - traditional path finding algorithms are presented to choose from. Features like intermediate pivots and wall - based boundary space are added. The motive is to visualize the first principles of the path finding algorithms.*

Keywords- Grid space, Search space, Path space, Shortest distance, Pivot, Graph Algorithms

I. INTRODUCTION

Visualizers, in abstraction, need not always visualize an algorithm, fitting a deterministic data set. In fact, there is no such primary data set at all. Instead, it can just be a set of rules to be followed. The stimulator can then follow these logical rules, bound to a particular behaviour, and produce the end state.

It augments our intellect by leveraging our visual system. Hence, abstract concepts can be grasped better, and we might be able to see other stuff too.

In principle, to find the shortest path between two vertices, a path finding algorithm, starts from the source, explores adjacent nodes until the destination vertex is found, after which the cheapest among which is chosen. This can be done with traditional algorithms like - A*, DFS, BFS, Dijkstra's and others.

Two primary problems of pathfinding are

1. To find a path between two nodes in a graph
2. The shortest path problem - to find the optimal shortest path.

Trivial algorithms such as depth-first and breadth-first search address the problem of finding a path between two nodes of a graph by exhausting all possibilities; starting from the given node, they iterate over all potential paths until they reach the destination node. The interface provided is abstract enough to simulate and visualize one of these algorithms to fetch an optimum path space.

II. GENERAL GRAPH SEARCH

Consider a general graph algorithm like Breadth-first search where:

Input: A graph G, starting vertex and ending vertex Output: In an undirected grid space algorithm returns the shortest path.

1. function BFS(G, root) is
2. Declare Q to be a queue
3. Initialize the starting vertex(root) as explored Page | 1
4. Q.enqueue(root)
5. while Q is not empty do
6. current_vertex := Q.dequeue()
7. if current_vertex is the goal then
8. return current_vertex
9. for all edges from current_vertex to w in G.adjacentEdges(current_vertex) do
10. if w is not labelled as explored then
11. label w as explored
12. Q.enqueue(w)

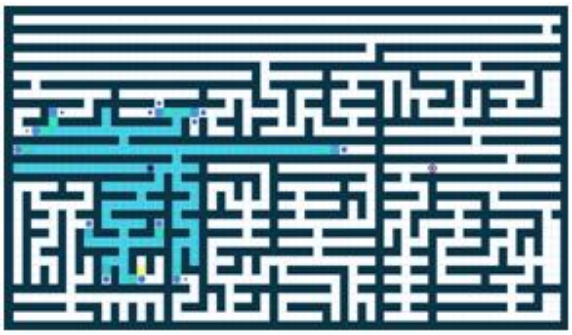

Fig. 1. Probing the grid space to find the most optimum path space (BFS)

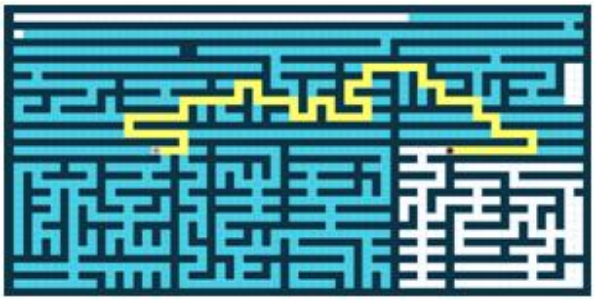

Fig. 2. Optimum path space (BFS)

Fig 1 and Fig 2 shows the pathfinding visualizations on a grid space. Visualizing the search space on a grid is easier than on a plain. After calculating and displaying the search space on the screen, the power to drag the end vertex to any point allows the user to visualize the search space and shortest path represented in yellow to change accordingly.

What we are proposing with our implementation is to make the user get a simple mental model of various path finding algorithms. This mental model will play a huge role in understanding why certain algorithms use different search spaces with different speeds.

Search spaces of every algorithm are different, the colour in blue represents the search spaces. A larger search space doesn't necessarily mean it returns the longest path, it only means that the algorithm took a longer route to compute the shortest path.

However, we understand that these facts about the algorithms are already demonstrated with mathematical proofs but gaining a mental model particularly on the search space and the variation of the search space for different algorithms for the same set of start and end vertices makes better sense to hold on to the idea.

III. CONCLUSION

We conclude by highlighting that from a development and a user point of view, a grid based visualization system for path finding algorithm makes sense.

This idea of visualizing in a grid system can be done for different non graph based simulations. Simulations like Monte Carlo are one good example which fits our analogy.

Since this is a niche topic, it's always good to have a different perspective which might lead to new discoveries.

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest and Clifford Stein, "Introduction to Algorithms", 1989.
[2] Jon Kleinberg and Eva Tardos, “Algorithm Design.”, 2005.
[3] Narsingh Deo, “Graph Theory with Applications to Engineering and Computer Science”, 1974.
[4] Kenneth H. Rosen, “Discrete mathematics and its applications”, 1984.
[5] Abhishek Kumar, Pramod Singh Rathore, and Vicente Garcia Diaz, "Swarm Intelligence Optimization: Algorithms and Applications", 2020