IMPROVE MINECRAFT PATHFINDING

STUDENTS: Rocky Zhenxiang Fang, Sisir Kadiveti, Arman Kazi, Hanze (Simon) Zhang

Problem Statement

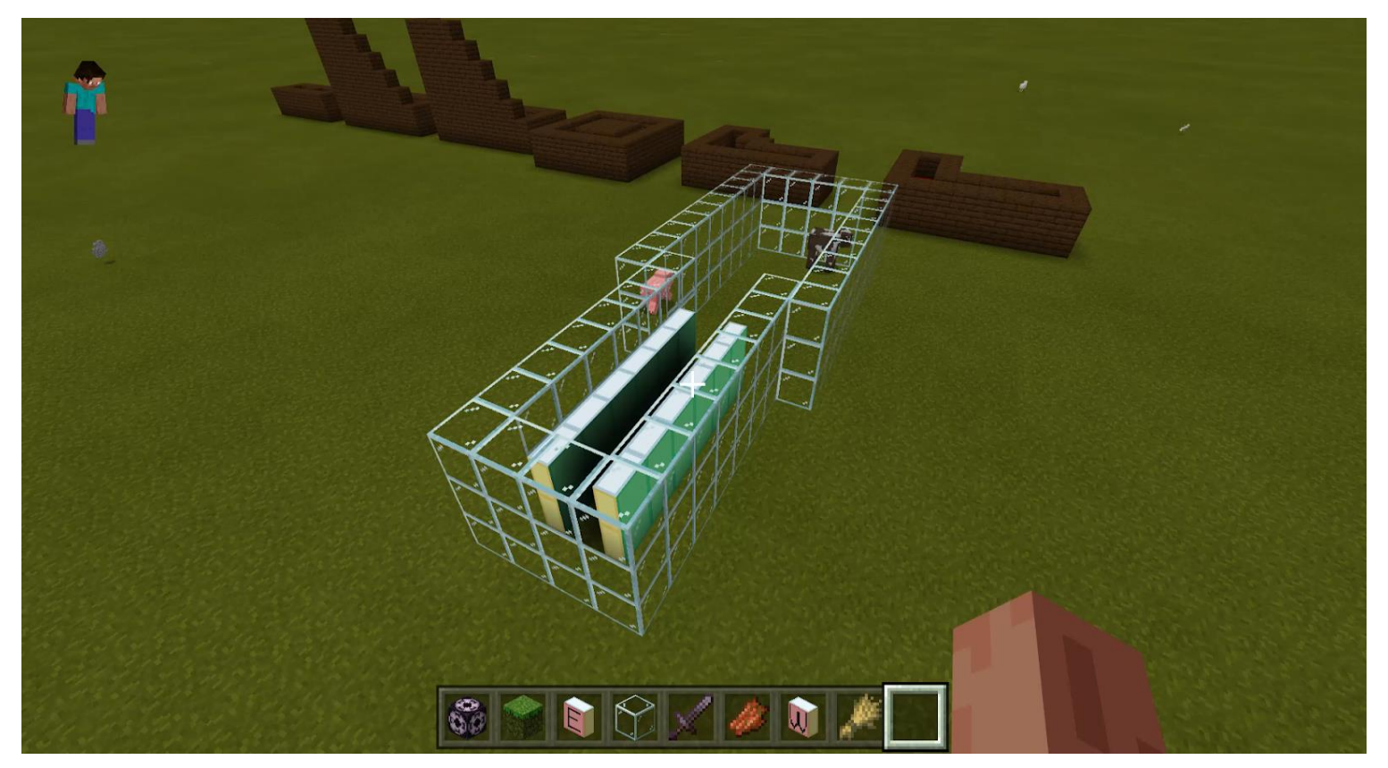
- In the world of Minecraft, there are many different types of mobs with distinct mob sizes and moving abilities.
- The current pathfinder in Minecraft has a very high time complexity in helping a mob with sizes larger than 1x1 find the optimal path to its target.
- In addition, Minecraft is limited in the types of blocks they can add due to limitations in the pathfinder. For example, vertical slabs create unique challenges that the existing blocks don't have.
- The current pathfinder in Minecraft cannot find the optimal path if the optimal path involves partial blocks.

Requirements

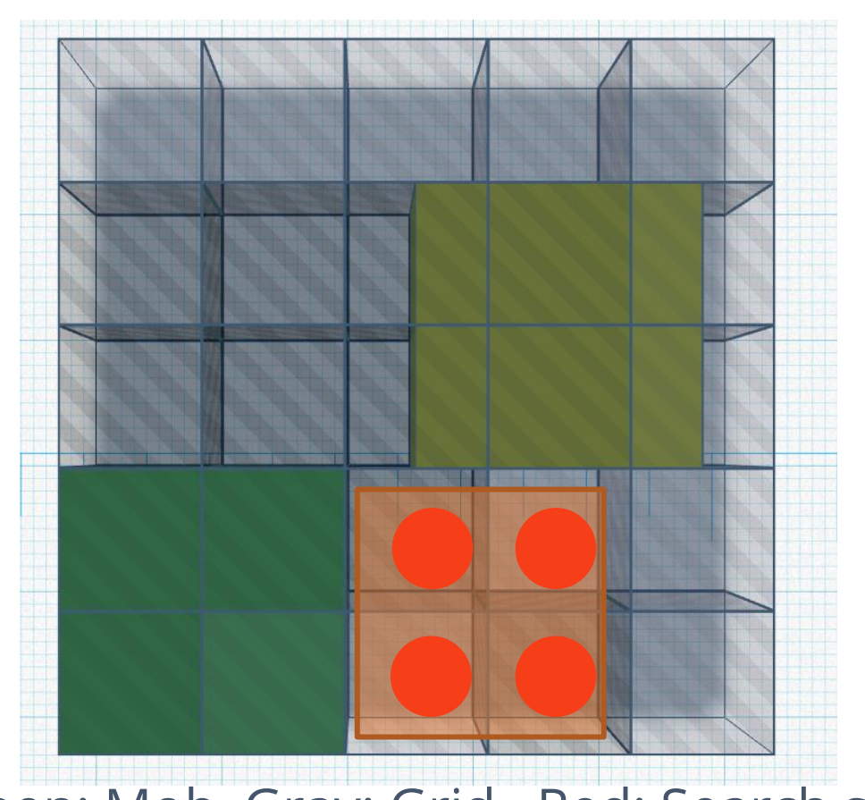
- The pathfinder should work with various mob size with partial blocks
- Analyze the performance difference between the original pathfinder and the new pathfinder.

Problem Analysis

- We have two problems to solve:
- 1. Mobs cannot path find through partial blocks.
- The reason behind this problem is that the pathfinder currently in Minecraft cannot recognize the space between two vertical slabs.



- For this example, when the cow is at one side of partial block tunnel, it will evaluate the feasibility of two partial blocks individually but will not combine the space in between them.

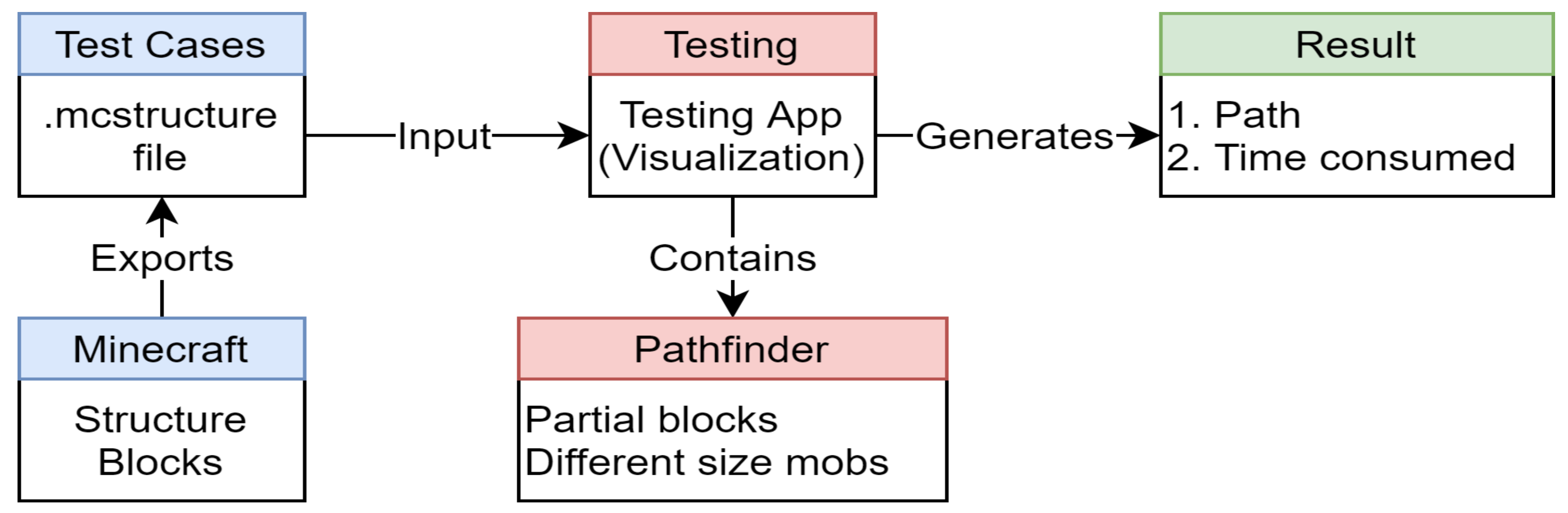


Green: Mob, Gray: Grid, Red: Search space
Yellow: Node, Orange: Newly generate node

- 2. It takes big mobs with a size larger than 1x1 a long time to find a path.
- The reason behind problem 2 is that the dimension of the search space of the current pathfinder is proportional to the mob sizes.
- For this example, every time the mob (green mob) wants to move right, it will need to check all red dots, which is proportional to its size
- Creating new search space (partial blocks) + cache the result (big mob) = Node system
- One node is presented in the graph above using yellow block
- Some nodes might not be aligned with the original grid so it can adapt to partial blocks.
- For searched space, we create a node to cache the result. In this example, if the red blocks are visited and cached in the node system, the next time it is visited, it will return the value from the cache but not recompute it.
- Nodes can be stored in the game file so the searching process will be faster

Testing Process

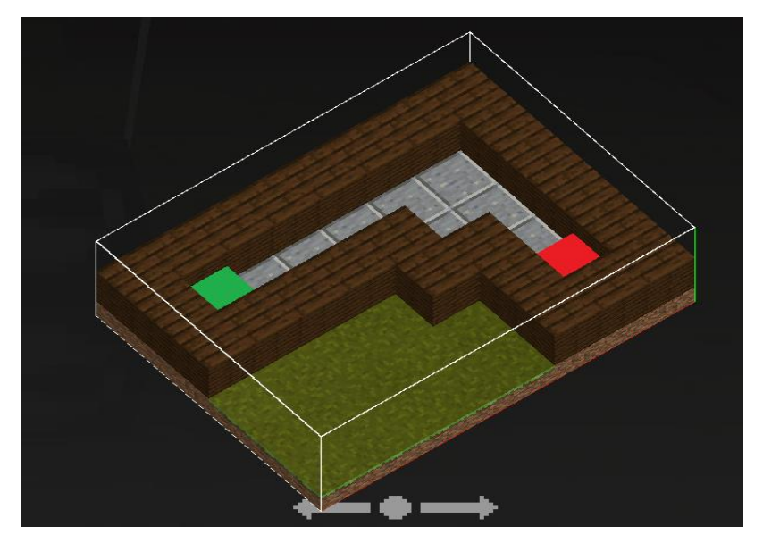
- Our testing process involved 3 processes:
- First, we use Minecraft to build our test cases and export them into a mcstructure file
- Second, we take the mcstructure file into our pathfinder and output the final path and time it spends
- Finally, we take the results and analysis them. For path, we want it to be the shortest and for the time, we want it to be spend a similar time to the original pathfinder

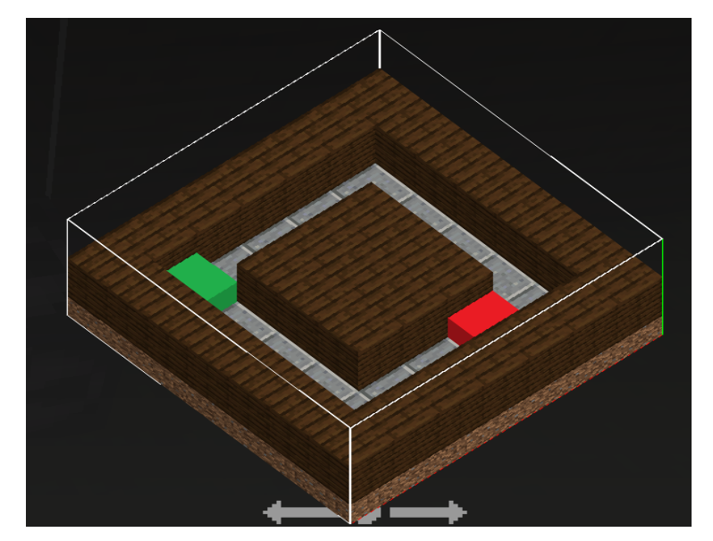


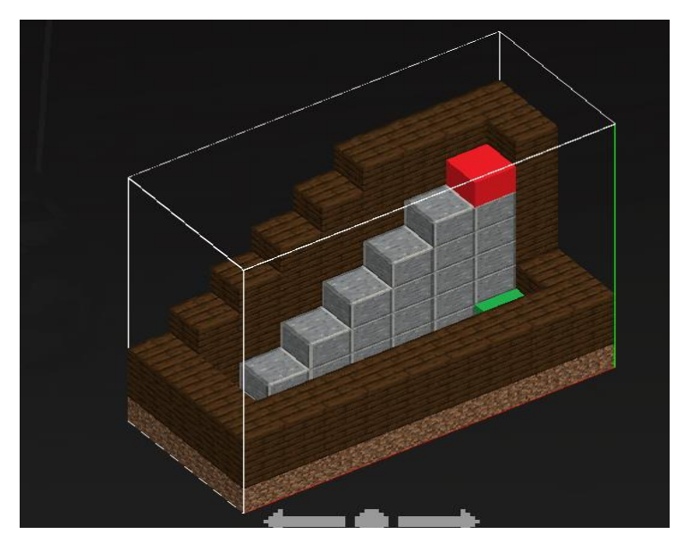
Test Cases

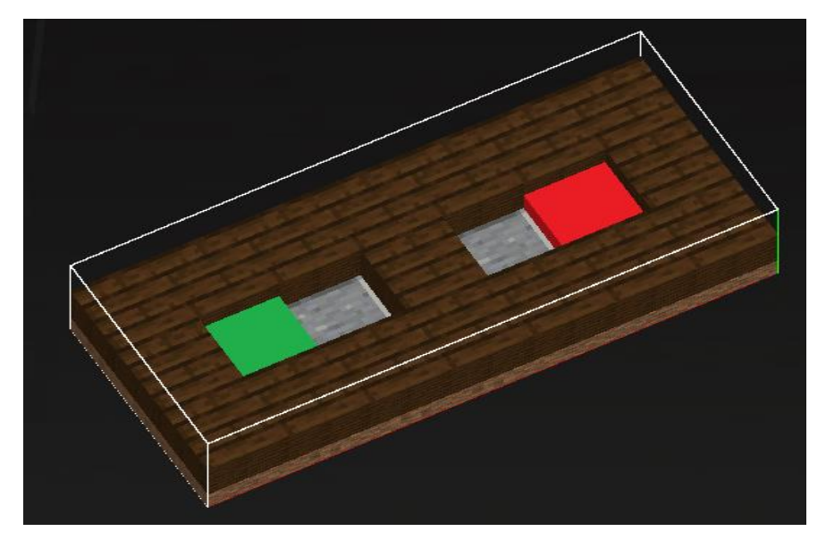
- We focus on the correctness and the performance of our pathfinder; thus, we design test cases to test them.
- For correctness, we determine it by looking at the path itself, it should support 8-directional movements, finding the shortest path, climbing up and sown in 3D space, and return the optimal partial path if necessary.
- For performance, we focus on what will happen if the distance between the origin and the target increase and different mob size under a different kind of world.
- Mob size, world size, and the blocks will change if the different kind of test cases are built.

Correctness

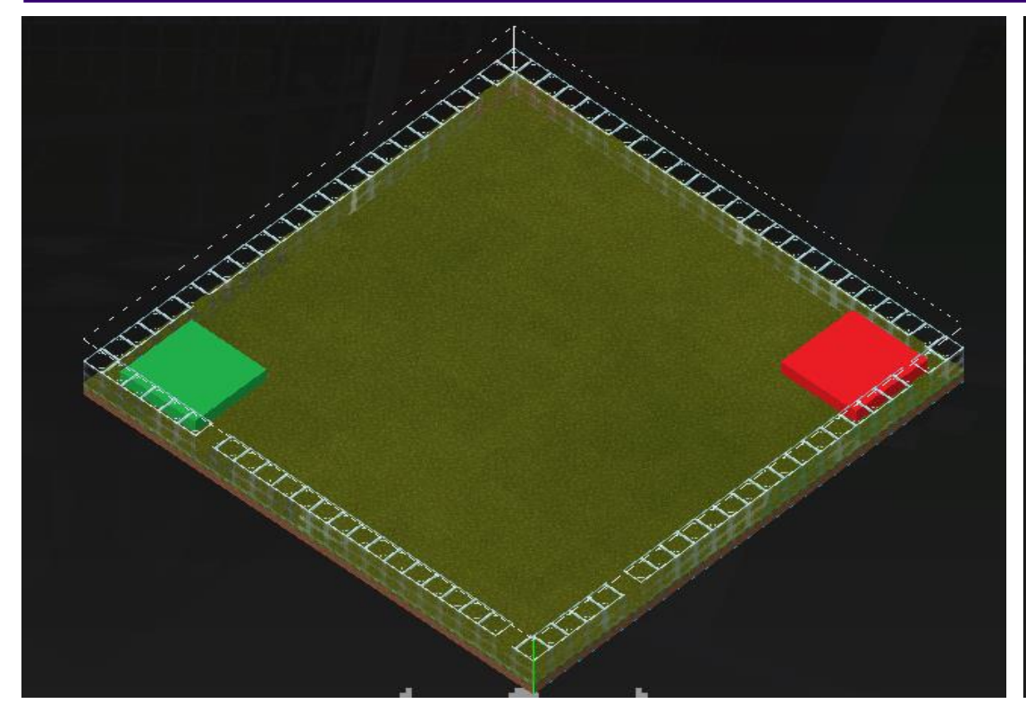

Diagonal Movements

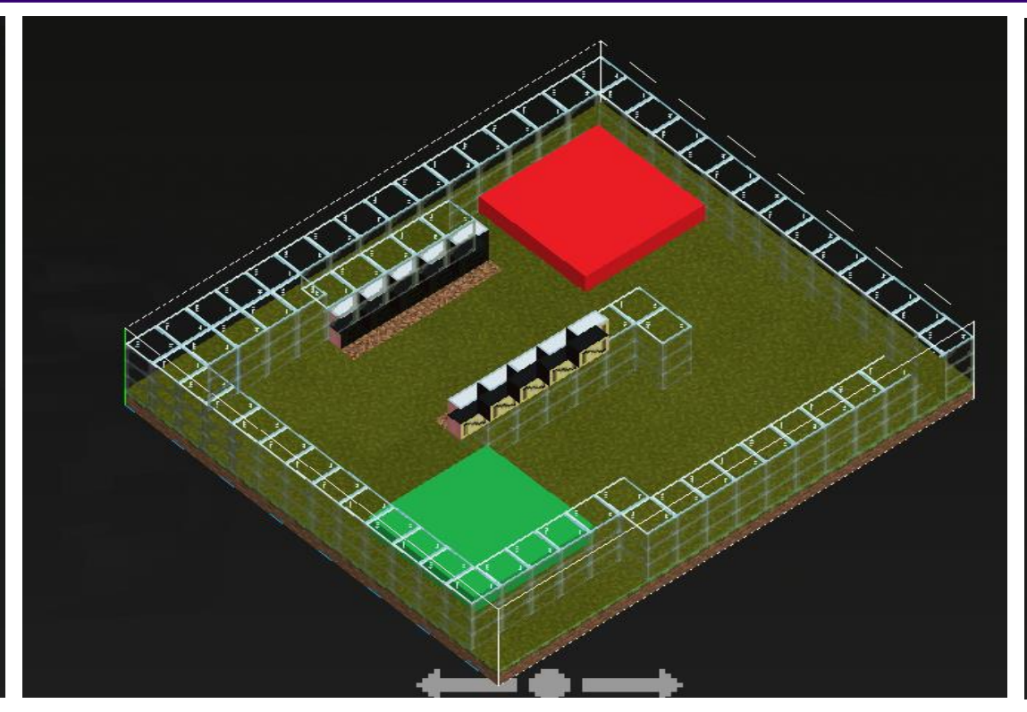

Shortest Path

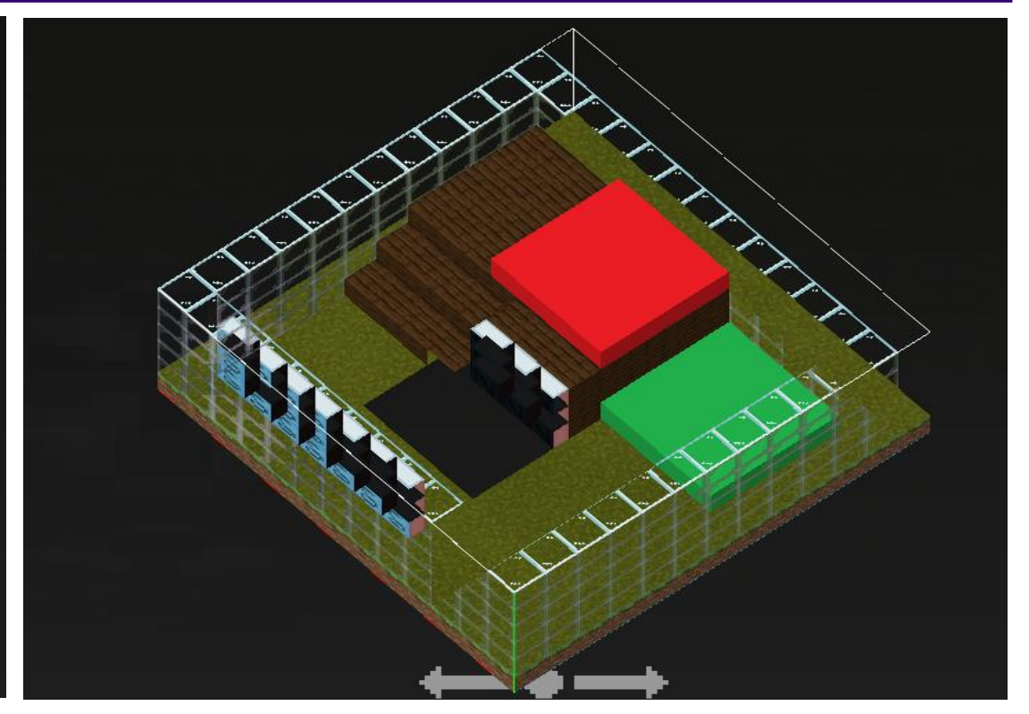

3D Space


Partial Path

Performance

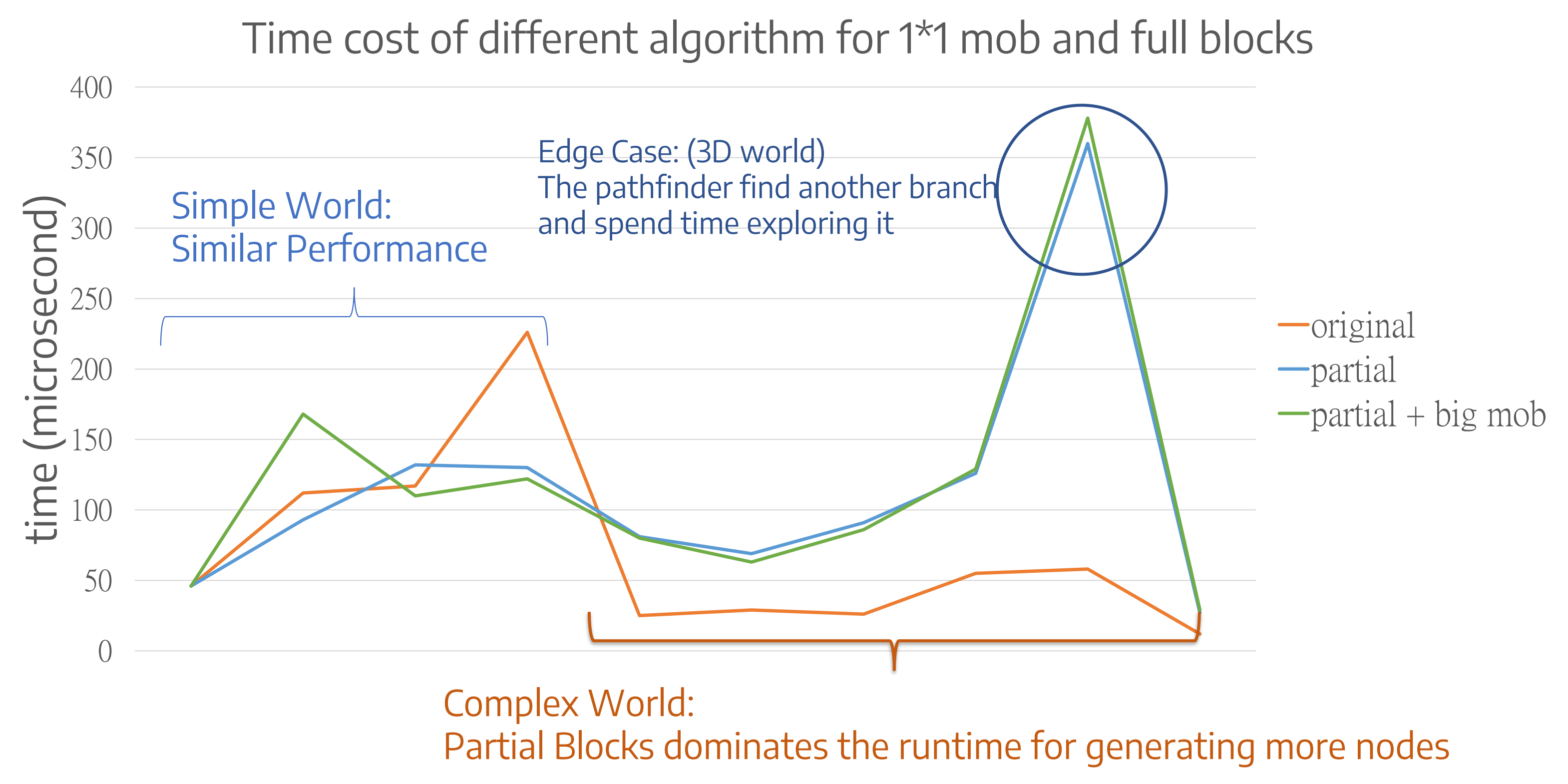

Simple World


Complex World

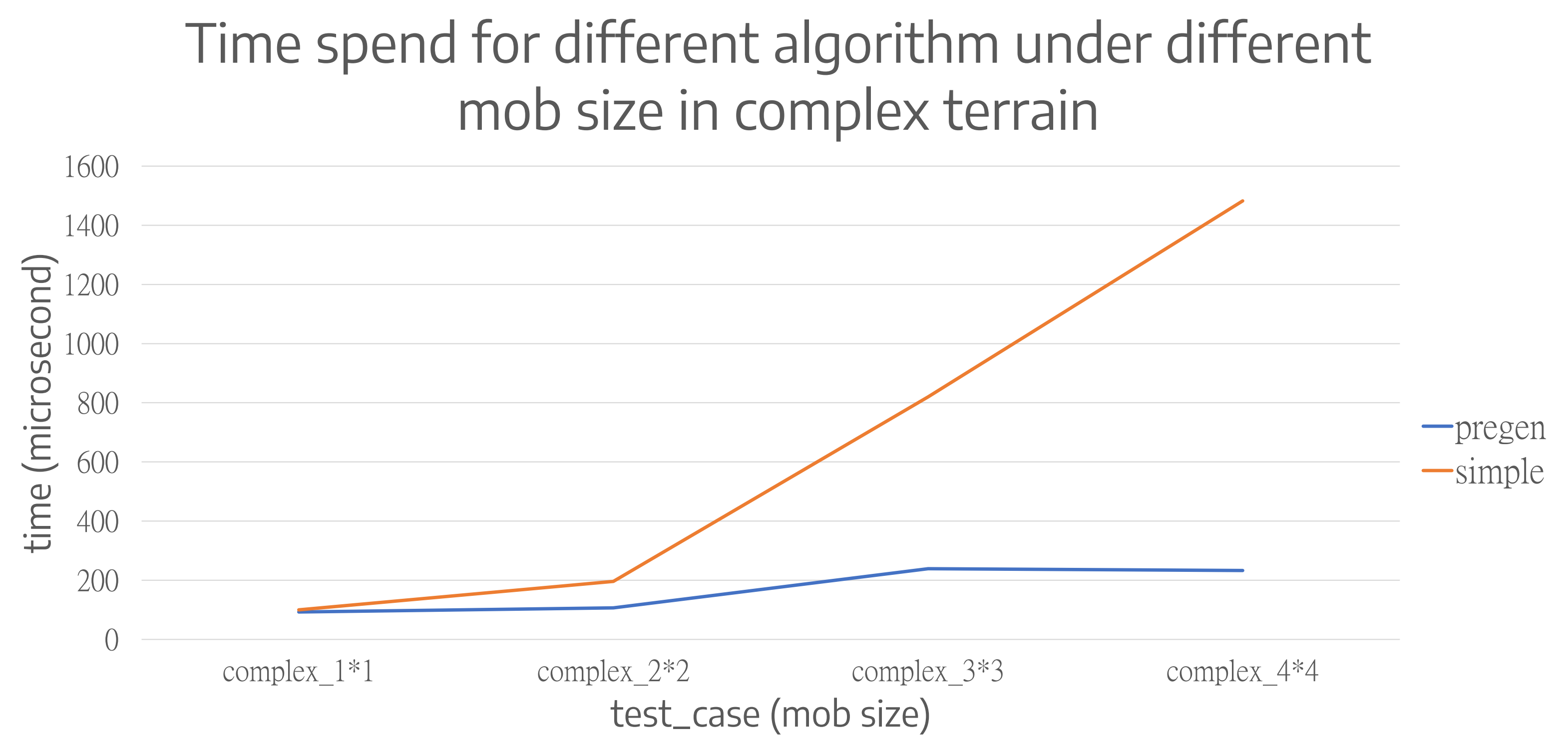

3D Complex World

Result

- Our result shows that our pathfinder will increase search time for adding partial blocks in some cases but will not be affected by world size and mob size.



- Simple worlds have different sizes but similar terrain, by only adding necessary nodes, the runtime does not change a lot.
- Complex world will open more partial nodes for potential paths that increase the runtime
- In some edge cases, the additional partial nodes might lead to a new path that is not the optimal path, which further increases the runtime.



- By caching the result, the runtime of the same world using different sized mob does not change compared to the original algorithm.

Conclusion and Future Work

- Big O run time is not affected if we generate all search nodes before pathfinding.
- Develop ways to change node in-game
- Develop ways to support arbitrary sized blocks and small mobs

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY *of* WASHINGTON

ADVISOR: Rania Hussein

INDUSTRY MENTOR: Jason Major

SPONSOR: Microsoft