**It’s the heuristic that matters**

It’s heuristics that matter and make the difference in searching spaces efficiently. For the Greedy Best First Search (GBFS) that was tested for this project, the heuristics were included within the getNeighbors function. As is typically done for this type of search, from the search algorithm, the getNeighbors function called and is sent the current position of the drone and a final or target position, then with simple math all of the next possible moves are determined. Based on the absolute value of the total distance in the respective plane, each move is ranked and put into a sorted list with the move having the smallest rank value, the least amount to go, at the front of the list. Since this simulation is three dimensional, twenty seven different possible moves were generated. Interestingly, the set of moves does include the determining and ranking of the Manhattan distance.

When approaching the search problem early in the project, a simple start from which to build was with the getNeighbors heuristic as a Best First Search (BFS). At that time, it was first calculating the best three dimensional move, as the optimal move, and thereafter, creating ranked based moves based in a single dimension. For example, since the parameters of the project are to move a single space in either or all of the X, Y, or Z planes, the best first move was a move in all three dimensions. Thereafter, those that followed in the list were more for a single dimension move and were ranked and added to the move list. Thus, there were only seven possible moves.

Then as the work progressed it was decided to evolve the simple getNeighbors heuristic function into a GBFS style search. The problem with the Greedy Best First Search algorithm is that the calculated moves may not be optimal. Also, as a warning, in talking about GBFS problems, [J3] notes: “Although GBFS is fundamental and powerful in planning, it has an essential drawback when heuristic functions return inaccurate estimates.” With this project, therefore, an optimal first move was calculated, set into the front of the queued list of moves, and tried first. If this move failed, the next move would be that of the ranked best moves determined by the getNeighbors function.

With these two versions of the getNeighbors function and heuristic, it was thought that there might be a positive difference between the use of the optimal move, always added to the front of the list, and not having the optimal move. That is, the moves would be calculated and ranked appropriately, but there would be no initial calculation of an optimal move. Trials were run and the results were not as expected. Shown in the graphs, the lines are identified in the legend. It is surprising that the GBFS with or without the optimum calculated value is a little slower than the Best First Search. It was thought that the BGFS with the fewer moves and better heuristics would compute faster. The two states for which these searches occurred are defined in Figures 1 and 2 below. World 1 is the initial state and World 2 in the final goal state. These searches were run on an Hewlett Packard laptop, running Windows 10 with an Intel i7-4510U CPU at 2.0 GHz, 12 Gigabytes of RAM, inside the JetBrains PyCharm Community Edition, 5.0.4, IDE.



0,3,4, drone

0,0,0, blue

0,0,4, green

1,0,0, red

1,0,4, green

2,0,0, purple

0,2,1, drone

0,1,4, blue

0,0,4, green

1,1,4, red

1,0,4, green

?,2,?, purple

Figure 2, World2

Figure 1, World1

The list in figure 2 shows that three cubes are required to be moved and the last cube, purple, was to be stacked upon another cube that had to be stacked on another. In other words, the purple cube is to be the third cube in a stack and that could be either on top of the blue or red cube. The data in the graph represent these runs. The final version of the getNeighbors function valued runs had a consistent count of twenty seven moves. Those of the early getNeighbors heuristic had a consistent value, or path length of 40.

The best explanation for the variations in processing time must be in background processes running and calculating of the more moves, twenty seven, used in the GBFS over the seven for the old. The real surprise, though, is in the graph when comparing the runs where the non-optimal value wasn’t used. Both of those sets of runs are faster than the GBFS with the optimal value.

**Future Work**

Considering the nature of this 3D project, future work has many interesting possibilities. Since one of our searches was a Greedy Best First Search, an interesting extension to this first project would be a test that is much like a simple game simulation. That is, to weight areas of the simulation and require the algorithm to determine best paths, either around or through, these zones to get to a goal position or build a cube tower. And/or if the drone approaches too closely, these zones could be populated with cubes that are a danger to the drone. The drone could then be penalized for coming too close to the cube in some way which could affect its ability to make it to the goal or create the final goal state.

Much like what was done in [J1] another future project would be the inclusion of one more or multiple drones each having an assigned goal. Although the system in [J1] was primarily two dimensional, the challenge could be considerable. That is, the system would then need to plan and coordinate multiple movements and, possibly, drone cube movements with prioritizations based on the respective assigned goals of each drone or the ultimate final state.

Although it would be quite an undertaking, another future project would be taking into account real world physical drones like done in [J2]. Although the drone in [J2] was a fixed wing craft and constantly moving, the our helicopter-like drone would be assigned physical parameters and the system would have to consider those aspects as the drone flies around the simulation – especially where a turn into an opening would require the consideration of the kinematics of the drone.

[J1] Wenjie Wang, Wooi Boon Goh, “Spatio-Temporal A\* Algorithms for Offline Multiple Mobile Robot Path Planning (Abstract),” Proceedings of 10th International Conference on Automated Agents and Multiagent Systems, pp 1091-1092, 2011.

[J2] Myung Hwangbo, James Kuffner, Takeo Kanade, “Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAV’s,” IEEE International Conference on Robotics and Automation, 2007.

[J3] Tatsuya Imai, Akihiro Kishimoto, “A Novel Technique for Avoiding Plateaus of Greedy Best-First Search in Satisficing Planning,” Proceedings, The Fourth International Symposium on Combinatorial Search, AAAI, 2011.