An Analysis of "Reusing Previously Found A* Paths for Fast Goal-Directed Navigation in Dynamic Terrain"

Wyatt Kormick
kormi001@umn.edu

February 11, 2018

Summary

A* is a decent informed search algorithm. Given consistent, useful heuristics it has no problem finding an optimal path. The problem with it is that, in a dynamic environment, it can be slow. It must be repeated every time a cost changes. Repeated A* recalculates an entirely new optimal path every time. In this paper, the authors suggest using information gleaned from the initial A* search in subsequent searches after a change in environment. Since the initial A* search found the optimal path, we can spend our computation time attempting to reconnect to the location where our optimal path got cut off, provided it is still optimal to do so. It also attempts to update the heuristics of previous nodes with the actual optimal cost using the previous searches' calculations, and if it cannot, it will update the heuristics to reestablish consistency after the environment change. This way, it can be proven that their algorithm, dubbed "Multipath Generalized Adaptive A*" (MPGAA*), preserved consistency, soundness, and optimality.

The authors' tested this MPGAA* algorithm and several other similar ones in the navigation of three different types of dynamic terrain. In each of these three situations, the algorithms had to reach a goal point. Every time the algorithm made k moves, cr/2 blocked cells became unblocked, and the same amount of unblocked cells became blocked. The algorithms were each tested with several different values of k and cr. On average, it was found that MPGAA* outperformed the other algorithms. There were several cases where a search algorithm called "D* Lite" outperformed MPGAA*.

Environment Analysis

The goal was to create a navigation algorithm for an agent moving through dynamic terrain, where the previously calculated path to the objective could possibly become blocked as the agent moved along it. While it was designed and experimented on for navigation purposes, it could also work in another sort of "dynamic terrain", that is instead multi-agent. One where the actions that an opposing agent takes blocks off certain future states momentarily. An example of this would be a chess AI. The AI could use the A* search to find a path to a desirable state, maybe not checkmate, but one that puts it in a favorable position. If the opponent were to then make a move that makes that path not possible, then the MPGAA* search could then find a new path to the desirable state.

While MPGAA* works well in dynamic terrain, it only does so when the environment is static, when the terrain changes after the algorithm makes a move. The algorithm would far less useful in a dynamic environment, where the terrain might change while the algorithm is searching for the path. It would end up with broken, incomplete, or suboptimal paths, or it would have to start the calculation over with every change.

Performance Versus Similar Algorithms

In their experimentation, the authors' main goal was to outperform D* Lite in terms of mean time to reach the goal. D* Lite is another search algorithm that works backwards from the goal to calculate the optimal path, and had so far been the fastest mean time to goal search algorithm. MPGAA* outperformed D* Lite in most tests. Where MPGAA* got slowed down was in a vastly dynamic environment, one where the environment changed often, and when it did, it changed a lot. D* Lite performed only about a fifth of a second better under such conditions, but in general was slower than MPGAA*. MPGAA* did not always outperform the "state-of-the-art" as such, but it did outperform most other algorithms in so many cases that it should be seriously considered in situations suited to it, in all but the most extreme dynamic environments.

Contribution

This paper shows that the simplest algorithms aren't always the fastest. Instead of running the same A* algorithm every time a new path is needed, the better way was to think about what information could be reused to remove redundant calculations. This algorithm dramatically speeds up the time it takes search for an optimal path to a goal, a process that is difficult to make efficient. In the future, algorithms could possible make use of the ideas presented here to use an informed search to find an optimal path even faster. As stated in the paper, there are already variations in the algorithm that find a possibly suboptimal path in dynamic terrain even faster than this algorithm.