CS1571 HW 1 Report

Tianjian Meng

9-30-2017

1. Heuristic functions
    1. For the Wireless sensor monitoring problem, the heuristic function I used is the number of targets which have not been assigned a sensor. Because the goal state is that all targets are monitored, this heuristic function could give the fastest way to assign each target a sensor. I think this function is admissible.
    2. For the Data aggregation problem, I used the time delay between the current node and its neighbours as heuristic function. I think this function is admissible because it is an optimal solution to a relaxed problem.
    3. For the Burnt pancake problem, I used the steps a pancake needs to get to the ideal state, or Manhattan distance, as the heuristic function. It is not admissible nor consistent heuristic because it would overestimate the cost to reach goal.
2. Best Search Strategy
    1. For the Wireless sensor monitoring problem, I think the best search strategy is Astar search. Although it would not always return the optimal solution, but it has a much lower time complexity and space complexity than unicost search. At the same time, its solution would be better than other search strategy at most cases.
    2. For Data aggregation problem, I think the best search strategy depends on the configuration. If the the depth of graph is large, the average degree of each node is not very small and the weight of paths doesn't differ a lot, bfs, unicost and iddfs would not work. For example, in the second test case, the time complexity and space complexity of bfs, unicost and iddfs would grow exponentially (the depth is 25 and the minimal node degree is 3, so both complexity would be at least 3^{24}), and the only feasible strategy is greedy search.
    3. For the Burnt pancake problem, the path cost would be the time we flip the pancakes, so actually the unicost search should have the same time and space complexity as bfs, which would grow exponentially. For the limited information (All five algorithms would not be able to

finish within the allotted time on the first and the second test case), I would say Astar search may be the best search strategy at least for this test case, but it could be a coincidence.

3. Couldn't finish within allotted time

    1. Only greedy algorithm could finish aggregation problem test case 2 in allotted time. I think the reason is that this graph is too complex. I did a little visualization on this graph (http://138.197.87.254:8888/cs1571vis), and found this graph is nearly impossible to find a path using most searching algorithms because the depth is 25 and the smallest node degree is 3. The search space is too large.
    2. All five algorithms couldn't finish pancake problem test case 1 and 2. I think the reason also exists in the large search space. At the same time, another reason is that I didn't find a good heuristic function to better estimate the remaining path cost.

4. Observation

    1. In my opinion, the unicost search and breath first search in the pancake problem should produce the same result, because the path cost for the pancake problem is the time we flip the pancakes, which is exactly same as the bfs doing. But in practice, it seems that unicost search would have about twice time and space to finish the search, I'm not sure whether it's just a coincidence. But both of them are between 6^8 and 6^9, so it may not be a big deal.
    2. I'm pretty surprised that iddfs performs pretty well in the pancake problem. It not only produce the optimal solution path, but also have a much better time and space complexity than bfs and unicost.
    3. For those algorithms couldn't finish within allotted time, I found that when I used "CTRL+C" to interrupt bfs, unicost, greedy and Astar, my terminal took a longer time to response to my operation. But when I interrupted iddfs, the response time is pretty short. I think this is because the space complexity of iddfs is much smaller than other algorithms, which means iddfs is more suitable for those problem with large search space than algorithms having relatively similar time complexity.