More Sophisticated Search

04 November 2020 21:19

Objectives:

- Understand that there are more sophisticated ways of searching
- Understand what Depth-First-Branch-And-Bound is
- Understand Iterative Deepening
- Become familiar with bi-directional search and island driven search

We can refine the aforementioned strategies using some more intelligent techniques

Depth-First Brand-And-Bound search

- Comines DFS with heuristic information
- Finds optimal solution most useful when there are multiple solutions and we want an optimal one
- Use the space of a DFS

- Suppose bound is the cost of the lowest cost path found to a goal so far
- What if the search encounters a path p such that the cost of p plus the heuristic applied to p is greater than the bound? - This means we can prune path p
- If we find a non-pruned path to the goal then we can set the bound equal to the cost of that path and then remember this as the best solution
- We should use DFS for linear space use
- We can guarantee an optimal solution with this algorithm

How do we initialise the bound? We can start it at infinity, or if we have an estimate we can use that

Notes on DFS BNB

- Cycle pruning works well with DFS BNB
- Multiple path pruning is not appropriate as storing explored set defeats space saving of dfs
- Can be combined with iterative deepening to increase the bound until either a solution is found or to show there is no solution

Context: Bounded Depth First Search

A bounded depth-first search takes a bound and does not expand paths that exceed the bound
Explores part of the search graph
Uses space linear in the depth of the search

The bound has to be the same or greater than the cost of getting to an optimal goal.
If we don't know that then we can do an iterative deepening search

Context: Iterative Deepening Search

1. Starts with a bound b=0
2. Do a bounded DFS with bound b
3. If a solution is found return that solution
4. Otherwise increment b and repeat

- This will find the same first solution as BFS
- Since using a depth-first search iterative deepening uses linear space
- Iterative Deepening has an asymptotic overhead of (b/(b-1)) times the cost of expanding the nodes at depth k using a BFS
- When b = 2 there is an overhead factor of 2, when b = 3 there is an overhead of 1.5 and as b

get higher the overhead factor reduces

Direction of Search

Bidirectional Search

- Search backward from the goal and forward from the start simultaneously
- This is effective since 2b^(k/2) < b^k and so can result in an exponential time saving
- The main problem is ensuring that the frontiers meet
- This is often used with one breadth-first method that builds a set of locations that can lead to the goal and in the other direction another method can be used to find a path to these interesting locations

Island Driven Search

- Find a set of islands between s and g
- There are m smaller problems rather than 1 big problem
- This can be effective since mb^(k/m) < b^k
- The problem is to identify the islands that the path must pass through it is difficult to guarantee optimality

Dynamic Programming

- For statically stored graphs build a table of dist(n), the actual distance of the shortest path from node n to a goal
- This can be built backwards from the goal
- This can be used locally to determine what to do

There are two main problems

- It requires enough space to store the graph
- The dist function needs to be recomputed for each goal