CMPSCI-683 Homework Assignment #1: Search

Patrick Pegus II

February 1, 2016

Problem 1. Iterative lengthening search is an iterative analogue of uniform-cost search. The basic idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.

1. Show that this algorithm is optimal for general path costs.

   Let modified uniform-cost path search (MUCS) be uniform-cost search (UCS) altered so that it does the following.

   (a) Discards any generated nodes exceeding the given limit.

   (b) Upon failure it returns the lowest path cost of any discarded node or failure if no node was discarded.

   The function signature of MUCS is:

```
function MUCS(problem, limit) return (solution, failure, cost)
end function
```

   Lemma 1. *MUCS is optimal.*

   Proof. Let p be the path returned by MUCS from start state S to goal state G. Let C^* be the optimal path cost from S to G. Now suppose that $cost(p) > C^*$. Since, like UCS, MUCS expands nodes in order of their optimal path cost, there is only one way this could have happened— MUCS must have discarded a generated node representing path p' that is an optimal path or a subpath of an optimal path $p' + p''$. But any node discarded by MUCS must have a path cost greater than the limit l. Since step costs are positive, $cost(p' + p'') > cost(p') > l >= cost(p) > C^*$. Therefore, the discarded node could not be an optimal path or a subpath of an optimal path. □

   Now define iterative lengthening search (ILS) as follows:

```
function ILS(problem) return (solution, failure)
    $s \leftarrow$ NULL
    $f \leftarrow$ NULL
    $c \leftarrow 0$
    while $s$ is NULL and $f$ is NULL do
```

```
        $s, f, c \leftarrow$ MUCS(problem, $c$)
    **end while return** $s, f$
**end function**
```

Theorem 1. *ILS is optimal.*

Proof. Since ILS only returns solutions returned by MUCS and MUCS is optimal, ILS is optimal. □

2. Consider a uniform tree with branching factor b, solution depth d, and unit step costs (each action costs one unit). How many iterations will iterative lengthening require?

   ILS will expand the start state in the first iteration, all states at depth 1 the second iteration, and all states at depth d at iteration $d + 1$.

3. Now consider the case where each step cost is a real number from the interval $[e, 1]$ for some $0 < e < 1$. How many iterations are required in the worst case? Try to derive the best estimate you can.

   A worst case example is each step cost along the path to the solution costs 1 and another infinitely long path exists having uniform step cost of e. In that case ILS will expand the start state the first iteration, all states reachable by a path of cost e the second iteration, $2e$ the third iteration, and the solution having path cost d at iteration $1 + \lceil \frac{d}{e} \rceil$.

Problem 2. Suppose you were doing an A* search and there were a number of different ways that you could compute an admissible heuristic value. Suppose that some were very cheap to compute but also very inaccurate estimators while others were very expensive to compute but were very accurate estimators. How would you determine which computation to use for a given application? Be as precise as possible. Describe intuitively a case in which a fast inaccurate heuristic would be beneficial and a case in which a more time-consuming accurate heuristic would be beneficial.

Let i be the fast inaccurate heuristic and a be the expensive accurate heuristic. We could say that the average time and space complexity of A* is $(1 + h)\, b^{1+\lceil \frac{C^*}{e} \rceil}$ where h is the average time to compute the heuristic of a given state, b is the effective branching factor achieved with the heuristic, C^* is the optimal solution cost, and e is the average step cost. Therefore, if C^* is much larger than e, b should be reduced as much as possible by choosing a. Alternatively, if C^* known to be not much greater than e, then i may be acceptable. However, if a can be cached and many searches must be computed within the domain, then the cost of computing the a cache can be amortized over all searches making the average time complexity near $b^{1+\lceil \frac{C^*}{e} \rceil}$.

Special properties of step cost function can also influence the heuristic choice. Suppose paths leading to an optimal solution have decreasing step costs and others do not. Then the search space would be self-pruning for A* even if the given heuristic was 0 for all states. Therefore, i would be a good choice, although a greedy search might be a better choice. Conversely, a would be required in the reverse situation.

Problem 3. Give an example of a search space in which A* would expand a node with a g-value that is greater than the cost of the optimal path from the initial state to that state. To do that, draw a small graph, designate one node as an initial state and one node as the goal state. Assign path costs to edges and an admissible heuristic estimate to each node.

1. Show the graph and trace the execution of A* until it terminates.

   In Figure 1, node representing path (I, C) with cost 5 is expanded before the node representing path (I, A, C) with cost 4. The execution trace is shown in Table 1.

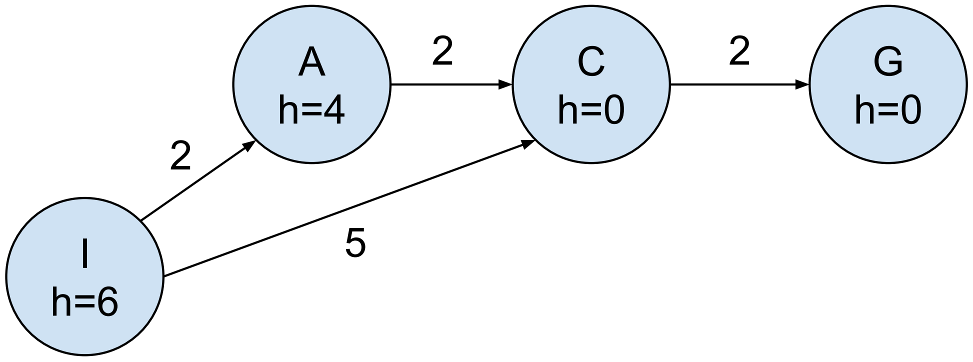

Figure 1: I and G are the initial and goal states, respectively.

Expansion	Open List
1	$[(I; 0+6)]$
2	$[(I, C; 5+0), (I, A; 2+4)]$
3	$[(I, A; 2+4), (I, C, G; 7+0)]$
4	$[(I, A, C; 4+0), (I, C, G; 7+0)]$
5	$[(I, A, C, G; 6+0), (I, C, G; 7+0)]$

Table 1: Execution trace of A* on the graph in Figure 1. Nodes are given symbolically as $(path; g+h)$.

2. Is your heuristic function consistent? Is it generally possible to create such an example when the heuristic is consistent? Explain your answer.

   My heuristic is inconsistent. $h(A) = 4 \nleq cost(A, C) + h(C) = 2$.