Artificial Intelligence Homework, Spring 2017

Turcu Gabriel-Virgil

Computer Science English Section, Second Year, Group 10206B

1 Problem statement

The task is to develop an application to evaluate the performance of an heuristic search algorithm. We consider Recursive best-first search RBFS. In what follows we present the description: This algorithm uses the f limit variable to keep track of the f-value of the best alternative path available from any ancestor of the current node. If the current node exceeds this limit, the recursion unwinds back to the alternative path. As the recursion unwinds, RBFS replaces the f-value of each node along the path with a backed-up value the best f-value of its children. In this way, RBFS remembers the f-value of the best leaf in the forgotten subtree and can therefore decide whether its worth reexpanding the subtree at some later time.

P2) Top-down proof with SLD resolution. Here you can vary the number n of propositional symbols, as well as the number k of rules.

Being given different rules that we can see as lines in a graph, we have to see if all our nodes from the starting nodes list have a path that leads to the "Yes" node. We will need to pick the branch that is most likely the fastest to give us an answer.

2 Pseudocode Section

Algorithm 1: Recursive best first search

```
Function RBFS(deque que)
    foreach vector in que do
        if the size of the vector is smaller than the minSize variable then
            the value of minSize is now the size of the vector
            the toSplit vector is now a copy of the vector
        end
    end
    foreach node in the vector toSplit do
        foreach vector in frontConnections[node] do
            if the size of the vector is 1 then
                we mark the element in that vector as being completed
            else
                we push the entire vector in the que
            end
        end
    end
    We delete the toSplit vector from the que.
    foreach node in toSplit do
        if the node is not marked as completed then
            ok=0
            break
        end
    end
    if ok then
        we mark the parent node as being completed
    end
    foreach node in toProve vector do
        if the node is not marked as completed then
            ok=0
            break
        end
    end
    if ok then
        we modify the value of programFinished
    end
    if programFinished = 1 then
        There is at least 1 path that leads to "yes" for our nodes in
         toProve.
    else
        There is no path that leads to "yes" for our nodes in toProve.
    end
1   return;
```

3 Application Design

- The heuristic way to pick the next nodes to split is by sorting all of the nodes by the number of parameters they have and pick the one with the fewest because that one is the one that is the most likely to finish quickest.
- The high level architectural overview of the application.
  - The application uses one recursive best first search function to solve the top-down proof with SLD resolution.
  - The main containers used in the function and outside the function are represented by :
    - frontConnections : this STL vector of vectors of int is used to hold connections leading downwards in our graph.
    - toProve : this vector is used to hold the nodes that will need to be proven as connected to the "yes" node.
    - backConnections : this vector is used to hold the parent for each node.
    - completed : this vector is 1 if the node can be marked as completed and 0 if it is not completed yet.
    - initialState : this deque that holds vectors is the initial state the RBFS will get at its first call.
- The Specification of the input.
- This is how the input will be given in the test file:
  1. First number represents the number of definitions. For each of the definitions we will have:
     1.1 The number of parameters.
     1.2 The starting node.
     1.3 All the parameters leading from the starting node.
  2. The number of nodes we need to prove.
  3. The nodes that we need to prove.
- The Specification of the output.

  The output is a message to the console that can be either:

  -The answer to the querry is true. This means there is at least 1 path that leads to "yes" for our nodes in toProve.

  -The answer to the querry is false. This means there is no path that leads to "yes" for our nodes in toProve.

4 Conclusions

4.1 Achievements

One thing I achieved during the project was understanding the logic behind some AI algorithms like RBFS and A* and how they pick their paths by estimating and guessing what is the best next path to chose without knowing too much in advance.

4.2 Challenging and interesting parts

One interesting part of the project was understanding the heuristic sorting for the nodes so we can pick the best one to split next.

4.3 Future directions for extending the project

One future direction for this project would be implementing it in a real world application or something like an AI in a game.

References

[1] Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms*. MIT Press, 3rd Edition, 2009.

[2] https://www.hackerrank.com/domains/ai/ai-introduction
http://artint.info/html/ArtInt.html
https://www.youtube.com/watch?v=EvvSYeI6BaQ
http://cs.gettysburg.edu/~tneller/papers/talks/RBFS_Example.htm
https://www.youtube.com/watch?v=tZYQywJJg9s