CSC 480 Artificial Intelligence

Brian Kwong

2025-04-18

Table of contents

1 Announcements

- Quiz 1 is due **tomorrow night** (Friday April 18th 2025)
- The Project Proposal is also due **tomorrow night** (Deadline has been extended 1 day)
  - This week submission will be a draft and ungraded. You will receive feedback from Professor Canaan
  - You then will have the opportunity to revise your project proposal whose final draft will be due **next Friday** (April 25th 2025). Will be **graded**

2 Search Algorithms

What makes search informed vrs one that is uninformed? Informed search has an **heuristics**

Heuristics A estimate of how close you are are to a end state/goal

2.0.1 Uninformed Search

In the previous lecture we have discussed two common tree search algorithms:

1. **Depth First Search** ALl child nodes of a particular branch are explored before moving onto the next branch.
2. **Breath First Search** All branches are explored at equal depths

> **Note**
>
> The primary advantage of DFS advantage is its **low memory** profile as only branch is needed to be stored in the system at a time, but but the derived solution may not be **optimal** and its not complete due to loops, or infinite number of branches.

For example in a infinite world game like `Minecraft` where the world is generate continually the search search becomes computationally infinite.

2.0.1.1 Depth Limited Depth First Search

- **Depth Limited Depth First Search** Keeps track of the current depth and stops once a maximum depth has been reached

```
depth_limited_dfs(root,max_d,d = 0)
  while d < max_d
    for child in root.children:
      depth_limited_dfs(child,max_d,d+1)

depth_limited_dfs(root,max_d)
```

2.0.1.1.1 Advantages of Depth limited DFS

- Optimal and complete if some solution up to n (where n is the maximum search depth)
- Many algorithms use depth limited DFS, such as the DeepBlue chess algorithm

2.0.1.1.2 Overhead of DFS & Depth Limited DFS

- Overhead of DFS comes from the fact you **continually** revisits the same nodes (parent nodes) as you explore in a bottom up approach
- There is also often similar paths that DFS will explore in depth which causes higher overhead.

> **Note**
>
> - The average **added** overhead is $\frac{1}{avg_branch_factor}$.
> - The total overhead is $1 + \frac{1}{avg_branch_factor}$.
>
> As the branching factor increase the approximation gets more exact (ie : sum of geometric sequences).