Artificial Intelligence

Lesson 2
Uninformed Search

- Uninformed search methods use only information available in the problem definition.
  - Breadth First Search (BFS)
  - Depth First Search (DFS)
  - Iterative DFS (IDA)
  - Bi-directional search
  - Uniform Cost Search (a.k.a. Dijkstra alg.)
Depth-First-Search Pseudo code

**DFS**(Graph graph, Node start, Vector Goals)
1. **L** ← make_stack(start)
2. While **L** not empty loop
   2.1 **n** ← **L**.remove_front()
   2.2 If goal (n) return true
   2.3 **S** ← successors (n)
   2.4 **L**.insert(S)
3. Return false
Depth-First-Search Attributes

- Completeness – No. Infinite loops or Infinite depth can occur.
- Optimality – No.
- Time Complexity: $O(b^m)$
- Memory Complexity: $O(bm)$
  - Where $b$ is branching factor and $m$ is the maximum depth of search tree

See water tanks example
Limited DFS Attributes

- Completeness – Yes, if $d \leq l$
- Optimality – No.
- Time Complexity: $O(b^l)$
  - If $d < l$, it is larger than in BFS
- Memory Complexity: $O(bl)$
  - Where $b$ is branching factor and $l$ is the depth limit.
Iterative-Deepening Attributes

• Completeness – Yes
• Optimality – yes, if graph is un-weighted.
• Time Complexity:

\[ O((d)b + (d - 1)b^2 + ... + (1)b^d) = O(b^d) \]

• Memory Complexity: \[ O(db) \]
  – Where \( b \) is branching factor and \( d \) is the maximum depth of search tree
Depth-First Iterative-Deepening

The numbers represent the order generated by DFID
Iterative deepening search $L=0$
Iterative deepening search $L=1$
Iterative deepening search $L=2$
Iterative Deepening Search $L=3$
Minimum cost path

• General minimum cost path-search problem:
  • Find shortest path from start state to one of the goal states in a weighted graph.
  • Path cost function is \( g(n) \): sum of weights from start state to goal.
Uniform Cost Search

• Also known as Dijkstra’s algorithm.
• Expand the node with the minimum path cost first.
• Implementation: priority queue.
Uniform Cost Search Attributes

- Completeness: yes, for positive weights
- Optimality: yes
- Time & Memory complexity:
  - Where $b$ is branching factor, $c$ is the optimal solution cost and $e$ is the minimum edge cost

\[ O(b^{\lceil c/e \rceil}) \]
Example of Uniform Cost Search

• Assume an example tree with different edge costs, represented by numbers next to the edges.

Notations for this example:
  — generated node
  — expanded node
Example of Uniform Cost Search

Closed list:

Open list:
Example of Uniform Cost Search

Closed list: a

Open list: b c
Example of Uniform Cost Search

Closed list: a, c

Open list: b, d, e

Costs: 2, 2, 3
Example of Uniform Cost Search

Closed list: a c b

Open list: d e f g

2 3 3 4
Example of Uniform Cost Search

Closed list: a c b d

Open list: e f g
Example of Uniform Cost Search

**Closed list:**

```
 a c b d e
```

**Open list:**

```
 f g
```

```
 3 4
```
Example of Uniform Cost Search

Closed list: a c b d e f

Open list: g 4
Example of Uniform Cost Search

Closed list:

Open list:
Exercise

• **Q:** Does a Uniform-Cost search be considered as a Best-First algorithm?
  
  • **A:** Yes. It can be considered as a Best-First algorithm with evaluation function \( f(n) = g(n) \).

• **Q:** In what scenarios IDS outperforms DFS?, BFS?
  
  • **A:**
    • IDS outperforms DFS when the search tree is a lot deeper than the solution depth.
    • IDS outperforms BFS when BFS run out of memory.

• **Q:** Why do we need a closed list?
  
  • **A:** Generally a closed list has two main functionalities:
    • Prevent re-exploring of nodes.
    • Hold solution path from start to goal (DFS based algorithms have it anyway).
מבחר 2014 מעודכן ב':

נתון מרחב החיפוש של וואקた W או קודקוד ההתחלה G- ו או קודקוד המטרה.
הצלוות המסומנות במחיר העלייה וה른ה המחיר למלטה נתונה ליד הקודקודים.

לכל אחה מתשיטה החיפוש
הבאות כתוב ציין האמ יוג
למטה, וביר, את כל הקודקודים
shmוצאים מה- OPEN List
лежаרות כל הפרפסים דיו
ה época - OPEN List נעשים
לפי סדר אוליפ-בייט.
כן כל בקט מינו, קודקודים בין
עコレיה יוסר בסדר אוליפ-בייט.