Artificial Intelligence

Lesson 3
Informed Search

• Incorporate additional measure of a potential of a specific state to reach the goal.

• A potential of a state to reach a goal is measured through a heuristic function $h(n)$, thus always $h(goal) = 0$.

• An evaluation function is denoted $f(n)$. 

Best First Search Algorithms

• Principle: Expand node \( n \) with the best evaluation function value \( f(n) \).
• Implement via a priority queue
• Algorithms differ with definition of \( f \):
  • Greedy Search: \( f(n) = h(n) \)
  • A*: \( f(n) = g(n) + h(n) \)
  • IDA*: iterative deepening version of A*
  • Etc’.
Properties of Heuristic functions

The 2 most important properties:

• relatively cheap to compute
• relatively accurate estimator of the cost to reach a goal. Usually a “good” heuristic is if $\frac{1}{2} \text{opt}(n) < h(n) \leq \text{opt}(n)$
• Examples:
  • Navigating in a network of roads from one location to another. **Heuristic function**: Airline distance.
  • Sliding-tile puzzles. **Heuristic function**: Manhattan distance - number of horizontal and vertical grid units each tile is displaced from its goal position.
Heuristic Function $h(n)$

- **Admissible/Underestimate**: $h(n)$ never overestimate the actual cost from $n$ to goal

- **Consistent/monotonic** (desirable) :
  
  $h(m) - h(n) \leq w(m,n)$ where $m$ is parent of $n$. This ensures $f(n) \geq f(m)$. 
Admissible/Underestimate

If a price estimate never exceeds the actual price from the current node to the target node.

The direct path (algorithm) from A to C will never exceed the true cost (the distance of the path A -> B -> C).
Consistent/monotonic

A function is said to be consistent (monotonic) on a set if for all points in the set, the value of the function is always increasing or always decreasing.

\[ y = x \]
Consistent/monotonic

If:

\[ h(n) - h(n') \leq C(n, n') \]
Example

Consider the heuristic such that \( h(n) = \) the minimum cost from a successor of \( n \) to one of its own successors, unless the successor is a goal node, in which case it is 0.
Best-FS Algorithm Pseudo code

1. Start with $open = [initial\text{-}state]$.  
2. While $open$ is not empty do  
   1. Pick the best node on $open$.  
   2. If it is the goal node then return with success. Otherwise find its successors.  
   3. Assign the successor nodes a score using the evaluation function and add the scored nodes to $open$
General Framework using Closed-list (Graph-Search)

**GraphSearch** (Graph graph, Node start, Vector goals)
1. $O \leftarrow \text{make\_data\_structure}(\text{start})$ // open list
2. $C \leftarrow \text{make\_hash\_table}$ // closed list
3. While $O$ not empty loop
   1. $n \leftarrow O.\text{remove\_front}()$
   2. If goal ($n$) return $n$
   3. If $n$ is found on $C$ → continue
   4. //otherwise
   5. $O \leftarrow \text{successors}\ (n)$
   6. $C \leftarrow n$
4. Return null //no goal found
Greedy Search Attributes

- Completeness: No. Inaccurate heuristics can cause loops (unless using a closed list), or entering an infinite path.
- Optimality: No. Inaccurate heuristics can lead to a non optimal solution.
- Time & Memory complexity:

\[ O(b^m) \]
A* Algorithm

• Combines greedy $h(n)$ and uniform cost $g(n)$ approaches.

• Evaluation function: $f(n)=g(n)+h(n)$
A* Pseudo code

A-Star(Graph graph, Node start, Node goal, HeuristicFunction h)
1. \( O \leftarrow \text{make\_priority\_queue}(\text{startNode}) \) // open list
2. \( C \leftarrow \text{make\_hash\_table} \) // closed list
3. While \( O \) not empty loop
   1. \( n \leftarrow O.\text{remove\_front}() \) // \( O \) is sorted by \( f(n) = g(n) + h(n) \) values
   2. If goal (n) return n
   3. If n is found on C \( \rightarrow \) continue
   4. //otherwise
   5. \( S \leftarrow \text{successors}(n) \)
   6. For each node s in S
      1. Set \( s.g = n.g + w(n,s) \)
      2. Set \( s.\text{parent} = n \) //for path extraction
      3. Set \( s.h = h(s) \) //to calculate \( f \)
      4. \( O \leftarrow s \)
   7. \( C \leftarrow n \)
4. Return null //no goal found
A* Algorithm (1)

• Completeness:
  • In a finite graph: Yes
  • In an infinite graph: if all edge costs are finite and have a minimum positive value, and all heuristic values are finite and non-negative.

• Optimality:
  • In tree-search: if \( h(n) \) is admissible
  • In graph-search: if it is also consistent
A* Algorithm (2)

• **optimally efficient**: A* expands the minimal number of nodes possible with any given (admissible and consistent) heuristic.

• Time and space complexity:
  • **Worst case**: Cost function \( f(n) = g(n) \)
    
    \[ O(b^{c/e}) \]

  • **Best case**: Cost function \( f(n) = g(n) + h^*(n) \)
    
    \[ O(bd) \]
IDA* Algorithm

• Each iteration is a depth-first search that keeps track of the cost evaluation $f = g + h$ of each node generated.

• The cost threshold is initialized to the heuristic of the initial state.

• If a node is generated whose cost exceeds the threshold for that iteration, its path is cut off.
IDA* Pseudo code

• **IDAS**tar-**Main** (Node root)
  1. Set bound = f(root);
  2. WHILE (bound<infinity)
     1. Set bound= IDAS**tar**(root, bound)

• **IDAS**tar(node n, Double bound)
  1. if n is a goal, Exit algorithm and return goal
  2. if n has no children, return infinity
  3. fn = infinity
  4. for each child c of n, Set f=f(c )
     1. IF (f<= bound) fn=min(fn, IDAS**tar**(c,bound))
     2. Else fn=min(fn,f)
  5. Return fn
IDA* Attributes

- The cost threshold increases in each iteration to the total cost of the lowest-cost node that was pruned during the previous iteration.
- The algorithm terminates when a goal state is reached whose total cost does not exceed the current threshold.
- Completeness and Optimality: Like A*
- Space complexity: $O(bd)$
- Time complexity*: $O(b^{c/e})$
Duplicate Pruning

• Do not enter the father of the current state
  • With or without using closed-list

• Using a closed-list, check the closed list before entering new nodes to the open list
  • Note: in A*, $h$ has to be consistent!
  • Do not remove the original check

• Using a stack, check the current branch and stack status before entering new nodes
שאלה 2: חיפוש

1. נתון מרחבי החיפוש הבא, ר-3 פונקציות הגרידנטיות:

א)ילי מהפונקציות
ב) conforme את הפתרון שלโคร עיווניה*A, לכל את מחויריסטיות
ג) מה המסלול שלโคร עיווניה Greedy אם ישנו ב- h1??
A - Çiir Marhav M[acement bo BFS iftah harba Fhtah Kódkódim.

DFS -

B - Çiir Marhav M[acement bo BFS iftah harba yot Kódkódim.

DFS -

BFS iftah M[acement bo A* iftah Fhtah Kódkódim bo A*.

DFS -

Iterative Depth Search iftah M[acement bo BFS iftah M[acement bo A*.

DFS -

Yftah (bosc héll) auto mescor shlo Kódkódim.