

Lecture 3: May 14th, 2018

*Lecturer: Alice Gao**Notes By: Harsh Mistry*

3.1 Search

Definition 3.1 *Propositional Search* : *Given a formula in propositional logic, determine if there is a way to assign truth values to the boolean variables to make the formula true*

We use search because we would like to find a solution when we are

- Not given an algorithm to solve a problem
- Given a specification of what a solution looks like
- Given costs associated with certain actions

Definition 3.2 *Search Problem* is defined by

- *A set of states*
- *A start state*
- *A goal state or goal test*
- *A successor function*
- *A cost associated with each action*

Definition 3.3 *Useful Terminology*

- ***Search Graph*** contains all the states and all the edges for the successor function
- ***Search Tree*** is constructed as we execute the algorithm
- ***Frontier*** contains all the leaf nodes available for expansion
- ***Expanding*** a nodes removes it form the frontier
- ***Generating*** a node adds the node to the frontier

3.1.1 Uninformed Search Algorithm

Uninformed search algorithms differ by the order in which we remove nodes from the frontier

- Breadth-first search treats the frontier as a queue (FIFO)
- Depth-first search treats the frontier as a stack (LIFO)

3.1.2 Informed Search

- Goal is to find the cheapest path from the start state to a goal state
- We can make use of two pieces of information
 - When we are at state n ,
 - $g(n)$: how far have we come from initial state to state n (cost from initial to n)
 - $h(n)$: heuristic (estimate) "Looking into future"
 - * How far to nearest goal
 - * Cheapest path to goal state

3.1.2.1 The Heuristic Function

- A search heuristic $h(n)$ is an estimate of the cost of the cheapest path from node n to a goal node
 - $h(n)$ is a arbitrary, non-negative (cost), and problem specific
 - if n is a goal node, $h(n) = 0$
 - $h(n)$ must be easy to compute without search. If it requires search, its a difficult problem

3.1.2.2 A* Search

- Uninformed and informed search algorithms
 - Treat (all 3 ways below as) the frontier as a priority queue order by $f(n)$
 - $f(n)$ should be the cost of a path
 - * Dijkstras algorithm (Lowest-Cost-First Search): $f(n) = g(n)$ - node where we travel the least
 - * Greedy search : $f(n) = h(n)$
 - * A* Search is a combination of Dijkstra and greedy search : $f(n) = g(n) + h(n)$

Algorithm 1 Search

```

1: let the frontier to be an empty list
2: add initial state to the frontier
3: while the frontier is not empty do
4:   remove curr_state from the frontier
5:   if curr_state is a goal state then
6:     return curr_state
7:   end if
8:   get all the successors of curr_state
9:   add all the successors to the frontier
10: end while
11: return no solution

```
