Informed Search

23 October 2020 21:44

Objectives:

- Best first search
- A* search
- Pruning search
- Depth bounded search
- Brand and bound search
- Heuristics

AIFCA 3.6-3.8

Uninformed search is generally very inefficient; if we have extra information about the problem we should use it
Improve search using problem specific knowledge
This is still offline problem solving since we have complete knowledge of the problem and solution

Heuristic

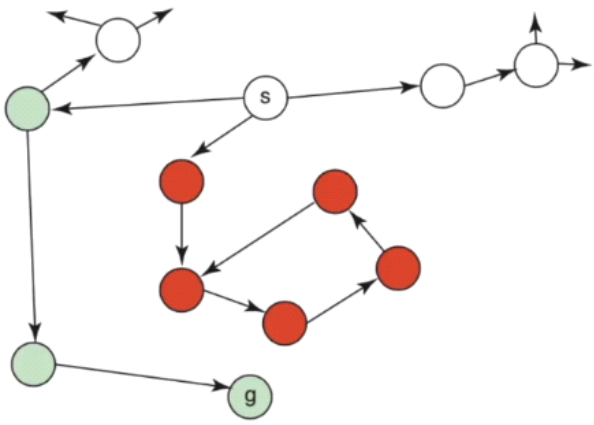
- idea: don't ignore the goal when selecting paths
- Often there is extra knowledge that can be used to guide the search: heuristics
- H(n) is an estimate of the cost of the shortest path from node n to a goal not
- H can be extended to paths
- H(n) is an underestimate if there is no path from n to a goal with cost strictly less than h(n)
- An admissible heuristic is a nonnegative heuristic function that is an underestimate of the actual cost of the path to the goal

Example Heuristic Function

- If the nodes are points on a Euclidean plane and the cost is the distance, h(n) can be the straight-line distance from n to the closest goal
- If the nodes are locations and cost is time, we can use the distance to a goal divided by max speed

Best-First Search Informed Search

- We can use the heuristic function to determine the order of the stack representing the frontier
- Idea: Select the path or node that is closet to a goal according to the heuristic function
- Heuristic depth-first search selects a neighbour so that the neighbour is selected first
- Greedy best fs selects a path on the frontier with the lowest heuristic value
- Best first search treats the frontier as priority
- queue ordered by h



The above is bad for a simple best-first search: heuristic depth first search will select the node below s and never terminate. Greedy best-first search will cycle between the nodes below s never finding an alternative route.

Complexity of Greedy best-first search

- The space complexity is exponential in path length b^n
- Time complexity is exponential in the path length
- Not guaranteed to find a solution, even if one exists
- It does not always find the shortest path

A* Search

It can be seen as an extension of Edsger Dijkstra's 1959 algorithm. A* achieves better performance by using heuristics to guide its search.

- Uses both path cost like in lowest-cost-first, and heuristic values, like in greedy best-first search
- Cost(p) is the cost of path p
- H(p) estimates the cost from the end of p to a goal
- Let(p) = cost(p) + h(p)
- F(p) estimates the total path cost of going from a start node to a goal via p

- A* is a mix of lowest-cost-first and best-first-search
- It treats the frontier as a priority queue ordered by f(p)
- It always selects the node on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that node.

A search algorithm is *admissible* if, whenever a solution exists, it returns an optimal solution

A* is admissible if

1. The arc costs are greater than 0
2. The branching factor is finite
3. The heuristic h is a non-negative and an underestimate

Why is it Admissible?

- If a path p to a goal is selected from a frontier, can there be a shorter path to a goal?
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p')$$

- Because h is an underestimate:

$$cost(p') + h(p') \leq cost(p'')$$

  for any path p'' to a goal that extends p'
- So $cost(p) \leq cost(p'')$ for any other path p'' to a goal.

- The frontier always contains the initial part of a path to a goal, before that goal is selected
- A* halts, as the costs of the paths on the frontier keeps increasing, and will eventually exceed any finite number
- Note admissibility does not guarantee that every node selected from the frontier is on an

optimal path
- Although it does ensure that the first solution found will be optimal, even in graphs with cycles

Iterative deepening A*

IDA* performs repeated depth-bounded depth-first searches. Instead of the bound being on the number of arcs in the path, it is a bound on the value of f(n). The threshold is initially the vlaue of f(s), where s is the start node. IDA* then carries out a depth-first depth-bounded search but never expands a path with a high f-value than the current bound.

How can a better heuristic function help?

- A* expands all paths from the start in the set
- A* also expands some paths from the set
- Increasing h while keeping it admissible reduces the size of the first of these sets
- If the second set is large there can be significant variability in the space and time of A*

Complexity of A*

Exponential time complexity
Exponential space complexity – you keep all nodes in memory

Strategy	Selection from Frontier	Path found	Space
Breadth-first	First node added	Fewest arcs	Exponential
Depth-first	Last node added	No	Linear
Iterative deepening	—	Fewest arcs	Linear
Greedy best-first	Minimal $h(p)$	No	Exponential
Lowest-cost-first	Minimal cost (p)	Least cost	Exponential
A^*	Minimal cost $(p) + h(p)$	Least cost	Exponential
IDA*	—	Least cost	Linear