The A* Search Algorithm

Siyang Chen

Introduction

A* (pronounced ‘A-star’) is a search algorithm that finds the shortest path between some nodes S and T in a graph.

Heuristic Functions

- Suppose we want to get to node T, and we are currently at node v. Informally, a *heuristic function* $h(v)$ is a function that 'estimates' how v is away from T.

Heuristic Functions

- Suppose we want to get to node T, and we are currently at node v. Informally, a *heuristic function* $h(v)$ is a function that 'estimates' how v is away from T.
- Example: Suppose I am driving from Durham to Raleigh. A heuristic function would tell me approximately how much longer I have to drive.

Admissible Heuristics

- A heuristic function is *admissible* if it never overestimates the distance to the goal.

Admissible Heuristics

- A heuristic function is *admissible* if it never overestimates the distance to the goal.
- Example: $h(v) = 0$ is an admissible heuristic.

Admissible Heuristics

- A heuristic function is *admissible* if it never overestimates the distance to the goal.
- Example: $h(v) = 0$ is an admissible heuristic.
- Less trivial example: If our nodes are points on the plane, then the straight-line distance $h(v) = \sqrt{(v_x - T_x)^2 + (v_y - T_y)^2}$ is an admissible heuristic.

Consistent Heuristics

- Suppose two nodes u and v are connected by an edge. A heuristic function h is *consistent* or *monotone* if it satisfies the following:

  $$h(u) \leq e(u, v) + h(v)$$

  where $e(u, v)$ is the edge distance from u to v.

Consistent Heuristics

- Suppose two nodes u and v are connected by an edge. A heuristic function h is *consistent* or *monotone* if it satisfies the following:

$$h(u) \leq e(u, v) + h(v)$$

  where $e(u, v)$ is the edge distance from u to v.
- Reasoning: If I want to reach T from u, then I can first go through v, then go to T from there. (This is very similar to the triangle inequality.)

Consistent Heuristics

- Suppose two nodes u and v are connected by an edge. A heuristic function h is *consistent* or *monotone* if it satisfies the following:
$$h(u) \leq e(u, v) + h(v)$$
where $e(u, v)$ is the edge distance from u to v.
- Reasoning: If I want to reach T from u, then I can first go through v, then go to T from there. (This is very similar to the triangle inequality.)
- Example: $h(v) = 0$ is a consistent heuristic.

Consistent Heuristics

- Suppose two nodes u and v are connected by an edge. A heuristic function h is *consistent* or *monotone* if it satisfies the following:

$$h(u) \leq e(u, v) + h(v)$$

  where $e(u, v)$ is the edge distance from u to v.
- Reasoning: If I want to reach T from u, then I can first go through v, then go to T from there. (This is very similar to the triangle inequality.)
- Example: $h(v) = 0$ is a consistent heuristic.
- Less trivial example, again: If our nodes are points on the plane, $h(v) = \sqrt{(v_x - T_x)^2 + (v_y - T_y)^2}$ is a consistent heuristic.

Consistent Heuristics

- Suppose two nodes u and v are connected by an edge. A heuristic function h is *consistent* or *monotone* if it satisfies the following:
$$h(u) \leq e(u, v) + h(v)$$
where $e(u, v)$ is the edge distance from u to v.
- Reasoning: If I want to reach T from u, then I can first go through v, then go to T from there. (This is very similar to the triangle inequality.)
- Example: $h(v) = 0$ is a consistent heuristic.
- Less trivial example, again: If our nodes are points on the plane, $h(v) = \sqrt{(v_x - T_x)^2 + (v_y - T_y)^2}$ is a consistent heuristic.
- All consistent heuristics are admissible. (Proof left to the reader.)

Description of A*

We are now ready to define the A* algorithm. Suppose we are given the following inputs:

- A graph $G = (V, E)$, with nonnegative edge distances $e(u, v)$
- A start node S and an end node T
- An admissible heuristic h

Let $d(v)$ store the best path distance from S to v that we have seen so far. Then we can think of $d(v) + h(v)$ as the estimate of the distance from S to v, then from v to T. Let Q be a queue of nodes, sorted by $d(v) + h(v)$.

Pseudocode for A*

$$d(v) \leftarrow \begin{cases} \infty & \text{if } v \neq S \\ 0 & \text{if } v = S \end{cases}$$

$Q :=$ the set of nodes in V, sorted by $d(v) + h(v)$

while Q not empty **do**
  $v \leftarrow Q.pop()$
  for all neighbours u of v **do**
    if $d(v) + e(v, u) \leq d(u)$ **then**
      $d(u) \leftarrow d(v) + e(v, u)$
    end if
  end for
end while

Comparison to Dijkstra's Algorithm

Observation: A* is very similar to Dijkstra's algorithm:

$$d(v) \leftarrow \begin{cases} \infty & \text{if } v \neq S \\ 0 & \text{if } v = S \end{cases}$$

$Q :=$ the set of nodes in V, sorted by $d(v)$
while Q not empty **do**
  $v \leftarrow Q.pop()$
  for all neighbours u of v **do**
    if $d(v) + e(v, u) \leq d(u)$ **then**
      $d(u) \leftarrow d(v) + e(v, u)$
    end if
  end for
end while

Comparison to Dijkstra's Algorithm

Observation: A* is very similar to Dijkstra's algorithm:

$$d(v) \leftarrow \begin{cases} \infty & \text{if } v \neq S \\ 0 & \text{if } v = S \end{cases}$$

$Q :=$ the set of nodes in V, sorted by $d(v)$
while Q not empty **do**
  $v \leftarrow Q.pop()$
  for all neighbours u of v **do**
    if $d(v) + e(v, u) \leq d(u)$ **then**
      $d(u) \leftarrow d(v) + e(v, u)$
    end if
  end for
end while

In fact, Dijkstra's algorithm is a special case of A*, when we set $h(v) = 0$ for all v.

Performance

How good is A*?

Performance

How good is A*?

- If we use an admissible heuristic, then A* returns the optimal path distance. Furthermore, any other algorithm using the same heuristic will expand at least as many nodes as A*.

Performance

How good is A*?

- If we use an admissible heuristic, then A* returns the optimal path distance. Furthermore, any other algorithm using the same heuristic will expand at least as many nodes as A*.
- In practice, if we have a consistent heuristic, then A* can be *much* faster than Dijkstra's algorithm.

Performance

How good is A*?

- If we use an admissible heuristic, then A* returns the optimal path distance. Furthermore, any other algorithm using the same heuristic will expand at least as many nodes as A*.
- In practice, if we have a consistent heuristic, then A* can be *much* faster than Dijkstra's algorithm.
- Example: Consider cities (points on the plane), with roads (edges) connecting them. Then the straight-line distance is a consistent heuristic.

Performance

How good is A*?

- If we use an admissible heuristic, then A* returns the optimal path distance. Furthermore, any other algorithm using the same heuristic will expand at least as many nodes as A*.
- In practice, if we have a consistent heuristic, then A* can be *much* faster than Dijkstra's algorithm.
- Example: Consider cities (points on the plane), with roads (edges) connecting them. Then the straight-line distance is a consistent heuristic.

(Proofs may be found in most introductory textbooks on artificial intelligence.)