Heuristic Search: A*

Alan Mackworth

UBC CS 322 - Search 4
January 16, 2013

Textbook §3.6
Recap

- Search heuristics: admissibility and examples
- Recap of BestFS
- Heuristic search: A*
Example for search with costs:
finding routes
Lowest-Cost First Search (LCFS)

- Expand the path with the lowest cost
  - Generalization of Breadth-First Search
  - Implemented as priority queue of cost values
- Only **complete** for strictly positive arc costs
  - Otherwise: a cycle with zero cost $\leq 0$ could be followed forever
- Only **optimal** for non-negative arc costs
  - Otherwise: a path that initially looks high-cost could end up getting a `refund`
- **Time and space complexity**: $\tilde{O}(b^m)$
  - E.g., uniform arc costs: identical to Breadth-First Search
Search heuristics

Def.: A search heuristic $h(n)$ is an estimate of the cost of the optimal (cheapest) path from node $n$ to a goal node.
Lecture Overview

• Recap

Search heuristics: admissibility and examples

• Recap of BestFS

• Heuristic search: A*
Last lecture’s example: finding routes

• What could we use as $h(n)$? E.g., the straight-line distance between source and goal node.
Admissibility of a heuristic

Def.: Let $c(n)$ denote the cost of the optimal path from node $n$ to any goal node. A search heuristic $h(n)$ is called admissible if $h(n) \leq c(n)$ for all nodes $n$, i.e. if for all nodes it is an underestimate of the cost to any goal.

- Example: is the straight-line distance admissible?
  
  **YES**
  
  - Yes! The shortest distance between two points is a line.
Admissibility of a heuristic

Def.:
Let $c(n)$ denote the cost of the optimal path from node $n$ to any goal node. A search heuristic $h(n)$ is called **admissible** if $h(n) \leq c(n)$ for all nodes $n$, i.e. if for all nodes it is an **underestimate** of the cost to any goal.

Another example: the goal is Urzizeni (red box), but all we know is the straight-line distances to Bucharest (green box).

- Possible $h(n) = \text{sld}(n, \text{Bucharest}) + \text{cost}(\text{Bucharest}, \text{Urzineni})$
- Admissible? **YES** **NO**
Example 2: grid world

- **Search problem**: robot has to find a route from start to goal location G on a grid with obstacles
- **Actions**: move up, down, left, right from tile to tile
- **Cost**: number of moves
- **Possible h(n)?**
  - Manhattan distance ($L_1$ distance) to the goal G: sum of the (absolute) difference of their coordinates
  - Admissible? YES or NO
Example 3: Eight Puzzle

• One possible $h(n)$:
  Number of Misplaced Tiles

• Is this heuristic admissible?
  YES  NO
Example 3: Eight Puzzle

- Another possible $h(n)$:
  Sum of number of moves between each tile's current position and its goal position

- Is this heuristic admissible?  
  
  YES  NO
How to Construct an Admissible Heuristic

• Identify relaxed version of the problem:
  – where one or more constraints have been dropped
  – problem with fewer restrictions on the actions

• Grid world: the agent can move through walls

• Driver: the agent can move straight

• 8 puzzle:
  - “number of misplaced tiles”: tiles can move everywhere and occupy same spot as others
  - “sum of moves between current and goal position”: tiles can occupy same spot as others

• Why does this lead to an admissible heuristic?
  - The problem only gets easier!
Lecture Overview

• Recap
• Search heuristics: admissibility and examples
• Recap of BestFS

Heuristic search: A*
A* Search

- A* search takes into account both
  - the cost of the path to a node $c(p)$
  - the heuristic value of that path $h(p)$.

- Let $f(p) = c(p) + h(p)$.
  - $f(p)$ is an estimate of the cost of a path from the start to a goal via $p$. 

![Diagram of A* search](image)
A* Search Algorithm

- A* combines elements of which two search algorithms?
  - Breadth-first
  - Depth-first
  - Best-first
  - Least cost first

- It treats the frontier as a priority queue ordered by $f(n)$

- It always chooses the path on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that path.

- Let’s see it in action:
A* in Infinite Mario Bros

http://www.youtube.com/watch?v=0s3d1LfjWCI
http://www.youtube.com/watch?v=DIkMs4ZHHr8
Analysis of A*

Def.: The time complexity of a search algorithm is the worst-case amount of time it will take to run, expressed in terms of
- maximum path length \( m \)
- maximum forward branching factor \( b \).

What is time complexity of A* in terms of \( m \) and \( b \)?

\[
\tilde{O}(b^m) \quad O(m^b) \quad O(bm) \quad O(b+m)
\]

- E.g., uniform costs and constant heuristic \( h(n) = 0 \)
  - Behaves exactly like BFS
A* completeness and optimality

• A* is complete (finds a solution, if one exists) and optimal (finds the optimal path to a goal) if:
  – the branching factor is finite
  – arc costs are $\epsilon > 0$
  – $h(n)$ is admissible -> an underestimate of the length of the shortest path from $n$ to a goal node.

• This property of A* is called admissibility of A*
Learning Goals for today’s class

- Construct heuristic functions for specific search problems
  Define/read/write/trace/debug different search algorithms
    - With/without cost
    - Informed/Uninformed
- Formally prove A* optimality (continued next class)