Lecture #36: Best-first State-space Search: A*

Objectives

- Understand Best-first search as a subclass of state-space search algorithms
- Introduce A* Search
- Compare and contrast with Branch & Bound
- Understand Admissible Heuristics

Acknowledgment

- Russell & Norvig: 4.1-4.2

Review: State-Space Search

- Basic idea:
  - Represent partial solutions (and solutions) as states
  - Find a solution by searching the state space
  - Search involves generating successors of states (i.e., "expanding" states) and pruning wisely
  - A search strategy is defined by specifying the following:
    - The manner of state expansion
    - The order of state expansion

Uninformed search strategies

- Use only the information available in the problem definition
- Are not informed by discoveries during the search.
- Examples:
  - Breadth-first search
  - Depth-first search
  - Depth-limited search
  - Iterative deepening search

Informed: Best-first search

- A type of informed search strategy
- Use an evaluation function \( f(s) \) for each state
  - Estimate of "desirability"
- Basic idea: Explore (expand) most desirable / promising state, according to the evaluation function
- Examples without agenda:
  - Greedy best-first search
- Examples with agenda:
  - Uniform-cost search
    - Single goal, State-space variant of Dijkstra's
  - A* Search
State-space search algorithms

Problem: Shortest Path

Admittedly, not a hard problem.

Map of Romania, distances in km

Greedy best-first search

- Evaluation function $f(n) = h(n)$ (heuristic)
  - estimate of cost from state $n$ to goal
- e.g., $h_{DLT}(n) =$ straight-line distance from $n$ to Bucharest
- Greedy best-first search expands the state that appears to be closest to goal
- No agenda
- Contrast with Simple Scissors (from proj. #4)

Greedy best-first search example

Greedy best-first search example

Greedy best-first search example
Greedy best-first search example

Properties of greedy best-first search
- **Complete?**
  - No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt → ...
  - or in dead ends
- **Optimal?**
  - No
- **Time?**
  - \( O(m*b) \)
- **Space?**
  - \( O(m) \)

A* search
- **Idea:** avoid expanding paths that are already too expensive (at best)
- **Very similar to Branch and Bound!**
  - But no BSSF
  - And no eager update of the BSSF
  - Solutions just go onto the agenda (always a priority queue)
  - First solution settled from the agenda wins
- **Also:** Split the bound function into two parts:
  - Evaluation function \( f(s) = g(s) + h(s) \)
  - \( g(s) \) = cost so far to reach state \( s \)
  - \( h(s) \) = estimated cost to go from state \( s \) to goal

A* search example

Analysis of Search Strategies
- We analyze search strategies in the following terms:
  - **Completeness:** does it always find a solution if one exists?
  - **Optimality:** does it always find an optimal solution, guaranteed?
  - **Time Efficiency:** number of states generated
  - **Space Efficiency:** maximum number of states in memory
  - “High water” mark on the agenda
- Time and space efficiency are measured in terms of:
  - \( b \): maximum branching factor of the search tree
  - \( d \): depth of the least-cost solution
  - \( m \): maximum depth of the state space (may be \( \infty \))
A* search example

Contrast with B&B.

Admissible heuristics

- A heuristic $h(s)$ is admissible iff for every state $s$, $h(s) \leq h^*(s)$
  - where $h^*(s)$ is the true cost to reach the goal state from $s$.
- An admissible heuristic never overestimates the cost to reach the goal
  - it is optimistic
  - it is a lower bound for minimization
  - (upper bound for maximization)
- Example: $h_{SLD}(s)$ for the shortest path problem
  - never overestimates the actual road distance
- **Theorem:** If $h(s)$ is admissible, then $A^*$ is optimal

Properties of $A^*$

- **Complete?**
  - Yes
  - unless there are infinitely many states $s$ such that $f(s) \leq f(G)$
- **Optimal?**
  - Yes, given an admissible heuristic
- **Time?**
  - $O(b^n)$, Exponential
- **Space?**
  - $O(b^n)$, worst case: keeps all states on agenda
For Comparison: Uniform Cost Search

- In state-space search: **Uniform cost search** is the single-goal version of Dijkstra's.
- How does A* compare with Dijkstra's algorithm?
  - Both use priority queue as agenda
  - A* solved shortest path (singular); Dijkstra's solves shortest paths (plural)
  - A* has better heuristic function; Dijkstra's: h(s)=0
    - It is uniformly the estimated remaining cost for every state
  - A* searches graph or state-space; Dijkstra's explores a graph
- How could you have used A* for the intelligent scissors project?
- What about the gene sequence alignment project?

For Project #7

- Would you use A* for Project #7?
- Why not?
- What happens if you use a simple priority queue as the agenda and the bound function as the priority in B&B?
- Would it be correct?
- Will you win?

For comparison: Branch and Bound Optimization

- B&B is **not** best-first!
- Idea: avoid expanding paths that are already too expensive at best, and prune them!
- BSSF = best solution so far
  - Allows pruning
  - Permits any-time solution
- Bound function f(s)
  - Estimated total cost of path through state s to goal (an optimistic bound)
- You know this well!

Properties of Branch and Bound

- **Complete?**
  - Yes
- **Optimal?**
  - Yes
  - As long as it runs to completion
  - As long as bound function is optimistic (a lower bound)
- **Time?**
  - \(O(b^n)\), Exponential
- **Space?**
  - At worst, keeps all states in memory and does not prune.
  - At best?

Assignment

- HW #27: A* for the 8-puzzle