Search: Advanced Topics and Conclusion

CPSC 322 – Search 6

Textbook §3.6
Lecture Overview

1. Recap
2. A* Analysis
3. Branch & Bound
4. A* Tricks

Search: Advanced Topics and Conclusion
A* Search

- A* search uses both path costs and heuristic values
  - $cost(p)$ is the cost of the path $p$.
  - $h(p)$ estimates the cost from the end of $p$ to a goal.

- Let $f(p) = cost(p) + h(p)$.
  - $f(p)$ estimates the total path cost of going from a start node to a goal via $p$.

- A* treats the frontier as a priority queue ordered by $f(p)$.
  - It always selects the node on the frontier with the lowest estimated total distance.
Lecture Overview

1. Recap
2. $A^*$ Analysis
3. Branch & Bound
4. $A^*$ Tricks
**A* is optimal**

**Theorem**

*If A* selects a path $p$, $p$ is the shortest (i.e., lowest-cost) path.*

- Assume for contradiction that some other path $p'$ is actually the shortest path to a goal.
- Consider the moment just before $p$ is chosen from the frontier. Some part of path $p'$ will also be on the frontier; let's call this partial path $p''$.
- Because $p$ was expanded before $p''$, $f(p) \leq f(p'')$.
- Because $p$ is a goal, $h(p) = 0$. Thus $cost(p) \leq cost(p'') + h(p'')$.
- Because $h$ is admissible, $cost(p'') + h(p'') \leq cost(p')$ for any path $p'$ to a goal that extends $p''$.
- Thus $cost(p) \leq cost(p')$ for any other path $p'$ to a goal. This contradicts our assumption that $p'$ is the shortest path.
$A^*$ is optimally efficient

- We can prove something even stronger about $A^*$: in a sense (given the particular heuristic that is available) no search algorithm could do better!

- **Optimal Efficiency:** Among all optimal algorithms that start from the same start node and use the same heuristic $h$, $A^*$ expands the minimal number of paths.
Lecture Overview

1. Recap
2. $A^*$ Analysis
3. Branch & Bound
4. $A^*$ Tricks

Search: Advanced Topics and Conclusion
Branch-and-Bound Search

- A search strategy often not covered in AI, but widely used in practice
- Depth-first: modest memory demands
- Uses a heuristic function: like $A^*$, can avoid expanding some unnecessary paths
  - in fact, some people see “branch and bound” as a broad family that includes $A^*$
  - these people would use the term “depth-first branch and bound”
Branch-and-Bound Search Algorithm

- Follow exactly the same search path as **depth-first search**
  - treat the frontier as a stack: expand the most-recently added path first
  - the order in which neighbors are expanded can be governed by some arbitrary node-ordering heuristic
- Keep track of a **lower bound** and **upper bound** on solution cost at each path
  - lower bound: $LB(p) = cost(p) + h(p)$
  - upper bound: $UB = cost(p')$, where $p'$ is the best solution found so far.
    - if no solution has been found yet, set the upper bound to $\infty$.
- When a path $p$ is selected for expansion:
  - if $LB(p) \geq UB$, remove $p$ from frontier without expanding it
    - this is called “pruning the search tree” (really!)
  - else expand $p$, adding all of its neighbours to the frontier
Branch and Bound Example

- [http://aispace.org/search/](http://aispace.org/search/)
- Example: Load from URL [http://cs.ubc.ca/~kevinlb/teaching/cs322/BnBSearchDemo.xml](http://cs.ubc.ca/~kevinlb/teaching/cs322/BnBSearchDemo.xml)
Branch-and-Bound Analysis

- Completeness: no, for the same reasons that DFS isn’t complete
  - however, for many problems of interest there are no infinite paths and no cycles
  - hence, for many problems B&B is complete
- Time complexity: $O(b^m)$
- Space complexity: $O(bm)$
  - Branch & Bound has the same space complexity as DFS
  - this is a big improvement over $A^*$!
- Optimality: yes.
Lecture Overview

1. Recap
2. A* Analysis
3. Branch & Bound
4. A* Tricks

Search: Advanced Topics and Conclusion
Other $A^*$ Enhancements

The main problem with $A^*$ is that it uses exponential space. Branch and bound was one way around this problem. Are there others?

- Iterative deepening
- Memory-bounded $A^*$
Iterative Deepening

- B & B can still get stuck in cycles
- Search depth-first, but to a fixed depth
  - set a maximum path length
  - augment branch and bound algorithm so that it also prunes paths that exceed the maximum length
  - if you don’t find a solution, increase the maximum path length and try again
- Counter-intuitively, the asymptotic complexity is not changed, even though we visit paths multiple times
Memory-bounded $A^*$

- Iterative deepening and B & B use a tiny amount of memory.
- What if we’ve got more memory to use?
- Keep as much of the fringe in memory as we can.
- If we have to delete something:
  - Delete the oldest paths.
  - “Back them up” to a common ancestor.