Nets and Optimal Search

—

Winston, Chapter 5

Michael Eisenberg and Gerhard Fischer
TA: Ann Eisenberg

AI Course, Fall 1997

Topics

global theme:

- to find not just one path to the goal (“main street”)
- but: the optimal path (“exploring side streets”)

Procedures:

- British Museum procedure
  - find all possible paths
  - use depth-first or breath-first search with modification: search continues until every solution is found

- procedures aspiring to do their work efficiently:
  - branch-and-bound
  - discrete dynamic programming
  - A*

Branch-and-Bound Search Expands the Least-Cost Partial Path

- eliminate unnecessary work

- the branch-and-bound scheme
  - keeps track of all partial path contending for further consideration
  - always extending the shortest path

- instead of terminating when a path is found, you terminate when the shortest partial path is longer than the shortest complete path

Branch-and-Bound Search

- To conduct a branch-and-bound search, form a one-element queue consisting of a zero-length path that contains only the root node.
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
  - Reject all new paths with loops.
  - **Add the remaining new paths, if any, to the queue.**
  - **Sort the entire queue by path length with least-cost paths in front.**
- If the goal node is found, announce success; otherwise, announce failure.

Adding Underestimates

· guesses about distances remaining

- **in general:**
  e(total path length) = d (already traveled) + e (distance remaining)
  known
  estimate

- **underestimates:**
  u (total path length) = u (already traveled) + u (distance remaining)
  known
  underestimate

- **for highway map:**
  - straight-line distance is guaranteed to be an underestimate
  - the crux of the approach:
    * underestimate as close as possible to the true distance
    * underestimate of close to zero is of little value

Branch-And-Bound Search With A Lower-Bound Estimate

- To conduct a branch-and-bound search with a lower-bound estimate, form a one-element queue consisting of a zero-length path that contains only the root node.

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
    - Reject all new paths with loops.
    - Add the remaining new paths, if any, to the queue.
    - Sort the entire queue **by the sum of the path length and a lower-bound estimate of the cost remaining**, with least-cost paths in front.

- If the goal node is found, announce success; otherwise, announce failure.

The Dynamic-Programming Principle

- **objective:** eliminate redundant partial paths

- **the Dynamic-Programming Principle**
  The best way through a particular, intermediate place is the

  best way to it from the starting place,

  +

  followed by the best way from it to the goal.

  There is no need to look at any other paths to or from the intermediate place.

Branch-and-Bound Search and Dynamic-Programming

- To conduct a branch-and-bound search with dynamic programming, form a one-element queue consisting of a zero-length path that contains only the root node.

- Until the first path in the queue terminates at the goal node or the queue is empty,
    - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
    - Reject all new paths with loops.
    - Add the remaining new paths, if any, to the queue.
    - **If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the minimum cost.**
    - Sort the entire queue by path length with least-cost paths in front.

- If the goal node is found, announce success; otherwise, announce failure.

A* Search

- To conduct A* search, form a one-element queue consisting of a zero-length path that contains only the root node.

- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.

  - Reject all new paths with loops.

  - **If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the minimum cost.**

  - **Sort the entire queue by the sum of the path length and a lower-bound estimate of the cost remaining, with least-cost paths in front.**

- If the goal node is found, announce success; otherwise, announce failure.

Several Search Procedures Find the Optimal Path

- The British Museum procedure is good only when the search tree is small.
- Branch-and-bound search is good when the tree is big and bad paths turn distinctly bad quickly.
- Branch-and-bound search with a guess is good when there is a good lower-bound estimate of the distance remaining to the goal.
- Dynamic programming is good when many paths converge on the same place.
- The A* procedure is good when both branch-and-bound search with a guess and dynamic programming are good.

Application: Obstacle Avoidance Problem for Robots

- **general idea (compare Number Scrabble and Tic-Tac-Toe):**
  redescribe the problem in a simpler representation
  solve the simpler problem
  redescribe the solution in the original representation

- **find a new presentation:**
  configuration-space obstacles
  turns: object-obstacle problems ---> point-obstacle problems
  allows to use A* search techniques

Find “One” Paths <----> Find the “Shortest” Path

- why do we care?

- in which situations do we care?

- **examples:**
  - main streets versus side streets (riding a bicycle to work, using a high functionality application)
  - buy a can of (low-altitude) tennis balls
  - find someone to repair a car
  - search in the WWW (40 000 items returned) ----> “optimal” solutions?

Summary

- The British Museum procedure is one of many search procedures oriented toward finding the shortest path between two points. The British Museum procedure relies on working out all possible paths.

- Branch-and-bound search **usually** saves a lot of time relative to the British Museum procedure. It works by extending the least-cost partial path until that path reaches the goal.

- Adding underestimates to branch-and-bound search improves efficiency. Deleting redundant partial paths, a form of dynamic programming, also improves efficiency. Adding underestimates and deleting redundant partial paths converts branch-and-bound search into A* search.

- The configuration-space transformation turns object-obstacle problems into point-obstacle problems. So transformed, robot path-planning problems succumb to A* search.