On-line exploration and navigation

Many problems in Algorithms and AI can be cast as searching a $\begin{pmatrix}\text{known}\\ \text{unknown}\end{pmatrix}$ environment $\begin{pmatrix}\text{of some}\\ \text{type}\end{pmatrix}$ in order to reach a $\begin{pmatrix}\text{known}\\ \text{unknown}\end{pmatrix}$ destination.

Yardsticks:

- Generality: what kinds of situations can you handle?
- What is your performance measure?

(As might expect, there's a tradeoff between generality and strength of performance measure)

Examples

Searching a known graph G for treasure t at an unknown location:

- Minimize $\max_t$[time-to-reach]: Traveling Salesman Problem.
- Given a prior P, minimize $\mathbf{E}_P$[time-to-reach]: Traveling Deliveryman problem.
- Minimize $\max_t$[time-to-reach/shortest-path]: general form of "cow-path" problem.

E.g., looking for a hole in a fence.

Searching for a hole in a fence

G is line from $-n$ to n. Start at position 0.

- Minimizing $\max_t$[time-to-reach].

  Go to one end and then to the other.

- Given a prior P, minimize $\mathbf{E}_P$[time-to-reach].

  Use dynamic programming. For each interval $[a, b]$ and each endpoint a or b, what is best way to visit the rest of the line?

- Minimize $\max_t$[time-to-reach/shortest-path].

  Optimum deterministic is:

  $$+1, -2, +4, -8, +16, \ldots$$

  Gets a competitive ratio of 9.

  Randomized on hwk...

  Related to combining online algorithms in bandit model.

Examples, contd

Searching an unknown graph G to reach a goal at unknown location:

- G is undirected; minimize distance as function of $|G|$: depth-first-search.
- G is undirected; minimize distance (roughly) as function of how many nodes BFS would visit: approximate BFS without teleportation.
- G is directed with probabilistic transitions; goal is to visit t as frequently as possible: Markov Decision Processes; Q-learning.
- G is directed, deterministic, but states not all distinguishable: learning a finite-state environment.
- Combine above two: POMDP problem.

Examples, contd, contd

Searching an unknown graph G for a goal at a *known* location.

- Makes sense if embed in a geometrical setting.
- Goal: do something better than brute-force exploration.

The Room Problem

- Start s at corner of n by n room. Room contains oriented rectangular obstacles of width at least 1.
- Goal is to reach t at center of room.

Note: Getting from t to s optimally is much easier.

Note: Brute-force search can be used to reach t in distance $O(n^2)$.

[hand-drawn versions of these next 5 slides contain pictures]

$O(n\sqrt{n})$-cost Algorithm Idea

- Go to temporary destination t' distance m by m from start.
- Make greedy down/right path.
- Make greedy down/left path back.
- If $m = \sqrt{n}$, we have paid cost $O(n)$ and removed $O(n \times \sqrt{n})$ area from consideration.

Analysis More Precisely

- Maintain invariant of a known monotone path from (x_0, n) to (n, y_0) cut by no obstacles.

  Begin with $x_0 = y_0 = 0$, path along room walls.

- At each iteration, increase x_0 or y_0 by at least $\sqrt{n}$ with only $O(n)$ cost.

Improvement to $n^{1+o(1)}$

- Apply recursively.
  - Let $T(n)$ be cost when $d(s,t) = n$.
  - Use $t' = (x_0 + m, y_0 + m)$.

$\Rightarrow$ Pay only $T(m) + O(n)$ to increase x_0 or y_0 by m.

- Repeat at most $2n/m$ times.

$$T(n) \le \frac{2n}{m}[T(m) + O(n)]$$

So

$$T(n) \le n \cdot n^{c/\sqrt{\log n}}.$$

Smoother tradeoff: get $O(n \log n)$

Related Settings

- No bounds on extents of obstacles. Look at ratio of on-line cost to shortest-path length, as a function of Euclidean distance n between s and t.
  - [PY][BRS] Deterministic $\Theta(\sqrt{n})$ bounds.
  - [BBFKRS] Randomized upper bound $O(n^{4/9})$.
  - [KRR] Randomized lower bound $\Omega(\log\log n)$. (Big gap!)
- What about improving performance with several trips?
  - [BC] In k trips can get ratio of $O(\sqrt{n/k})$.
- Partial information more generally?
- Navigating in a changing environment?