search is also helpful in nondeterministic domains because it allows the agent to focus its computational efforts on the contingencies that actually arise rather than those that *might* happen but probably won't.

Of course, there is a tradeoff: the more an agent plans ahead, the less often it will find itself up the creek without a paddle. In unknown environments, where the agent does not know what states exist or what its actions do, the agent must use its actions as experiments in order to learn about the environment.

A canonical example of online search is the **mapping problem**: a robot is placed in an unknown building and must explore to build a map that can later be used for getting from A to B. Methods for escaping from labyrinths—required knowledge for aspiring heroes of antiquity—are also examples of online search algorithms. Spatial exploration is not the only form of online exploration, however. Consider a newborn baby: it has many possible actions but knows the outcomes of none of them, and it has experienced only a few of the possible states that it can reach.

Mapping problem

4.5.1 Online search problems

An online search problem is solved by interleaving computation, sensing, and acting. We'll start by assuming a deterministic and fully observable environment (Chapter 17 relaxes these assumptions) and stipulate that the agent knows only the following:

- ACTIONS(s), the legal actions in state s;
- $c(s,a,s')$, the cost of applying action a in state s to arrive at state s'. Note that this cannot be used until the agent knows that s' is the outcome.
- IS-GOAL(s), the goal test.

Note in particular that the agent *cannot* determine RESULT(s,a) except by actually being in s and doing a. For example, in the maze problem shown in Figure 4.19, the agent does not know that going *Up* from (1,1) leads to (1,2); nor, having done that, does it know that going *Down* will take it back to (1,1). This degree of ignorance can be reduced in some applications—for example, a robot explorer might know how its movement actions work and be ignorant only of the locations of obstacles.

Finally, the agent might have access to an admissible heuristic function $h(s)$ that estimates the distance from the current state to a goal state. For example, in Figure 4.19, the agent might know the location of the goal and be able to use the Manhattan-distance heuristic (page 97).

Typically, the agent's objective is to reach a goal state while minimizing cost. (Another possible objective is simply to explore the entire environment.) The cost is the total path cost that the agent incurs as it travels. It is common to compare this cost with the path cost the agent would incur *if it knew the search space in advance*—that is, the optimal path in the known environment. In the language of online algorithms, this comparison is called the **competitive ratio**; we would like it to be as small as possible.

Competitive ratio

Online explorers are vulnerable to **dead ends**: states from which no goal state is reachable. If the agent doesn't know what each action does, it might execute an action such as "jump into a bottomless pit" and remain stuck there. In general, *no algorithm can avoid dead ends in all state spaces.* Consider the two dead-end state spaces in Figure 4.20(a). An online search algorithm that has visited states S and A cannot tell if it is in the top state space or the bottom one; the two look identical given what the agent has seen. Therefore, there is no

Dead end