AI

Homework assignment#2 (Chap4)

蔡宗諺 2020.12.5

4.1

4.1 Give the name of the algorithm that results from each of the following special cases:

a. Local beam search with $k = 1$.

b. Local beam search with one initial state and no limit on the number of states retained.

c. Simulated annealing with $T = 0$ at all times (and omitting the termination test).

d. Simulated annealing with $T = \infty$ at all times.

e. Genetic algorithm with population size $N = 1$.

Hint

- Breadth-first search
- Random-walk search
- First-choice hill-climbing search
- Hill-climbing search

4.6

4.6 Explain precisely how to modify the AND-OR-GRAPH-SEARCH algorithm to generate a cyclic plan if no acyclic plan exists. You will need to deal with three issues: labeling the plan steps so that a cyclic plan can point back to an earlier part of the plan, modifying OR-SEARCH so that it continues to look for acyclic plans after finding a cyclic plan, and augmenting the plan representation to indicate whether a plan is cyclic. Show how your algorithm works on (a) the slippery vacuum world, and (b) the slippery, erratic vacuum world. You might wish to use a computer implementation to check your results.

- generate a cyclic plan if no acyclic plan exists

```
function AND-OR-GRAPH-SEARCH(problem) returns a conditional plan, or failure
  OR-SEARCH(problem.INITIAL-STATE, problem, [])
```

```
function OR-SEARCH(state, problem, path) returns a conditional plan, or failure
  if problem.GOAL-TEST(state) then return the empty plan
  if state is on path then return failure
  for each action in problem.ACTIONS(state) do
      plan ← AND-SEARCH(RESULTS(state, action), problem, [state | path])
      if plan ≠ failure then return [action | plan]
  return failure
```

```
function AND-SEARCH(states, problem, path) returns a conditional plan, or failure
  for each s_i in states do
      plan_i ← OR-SEARCH(s_i, problem, path)
      if plan_i = failure then return failure
  return [if s_1 then plan_1 else if s_2 then plan_2 else ...if s_{n-1} then plan_{n-1} else plan_n]
```

Figure 4.11 An algorithm for searching AND–OR graphs generated by nondeterministic environments. It returns a conditional plan that reaches a goal state in all circumstances. (The notation $[x \mid l]$ refers to the list formed by adding object x to the front of list l.)

function AND-OR-GRAPH-SEARCH(*problem*) **returns** *a conditional plan, or failure*
  OR-SEARCH(*problem*.INITIAL-STATE, *problem*, [ ])

---

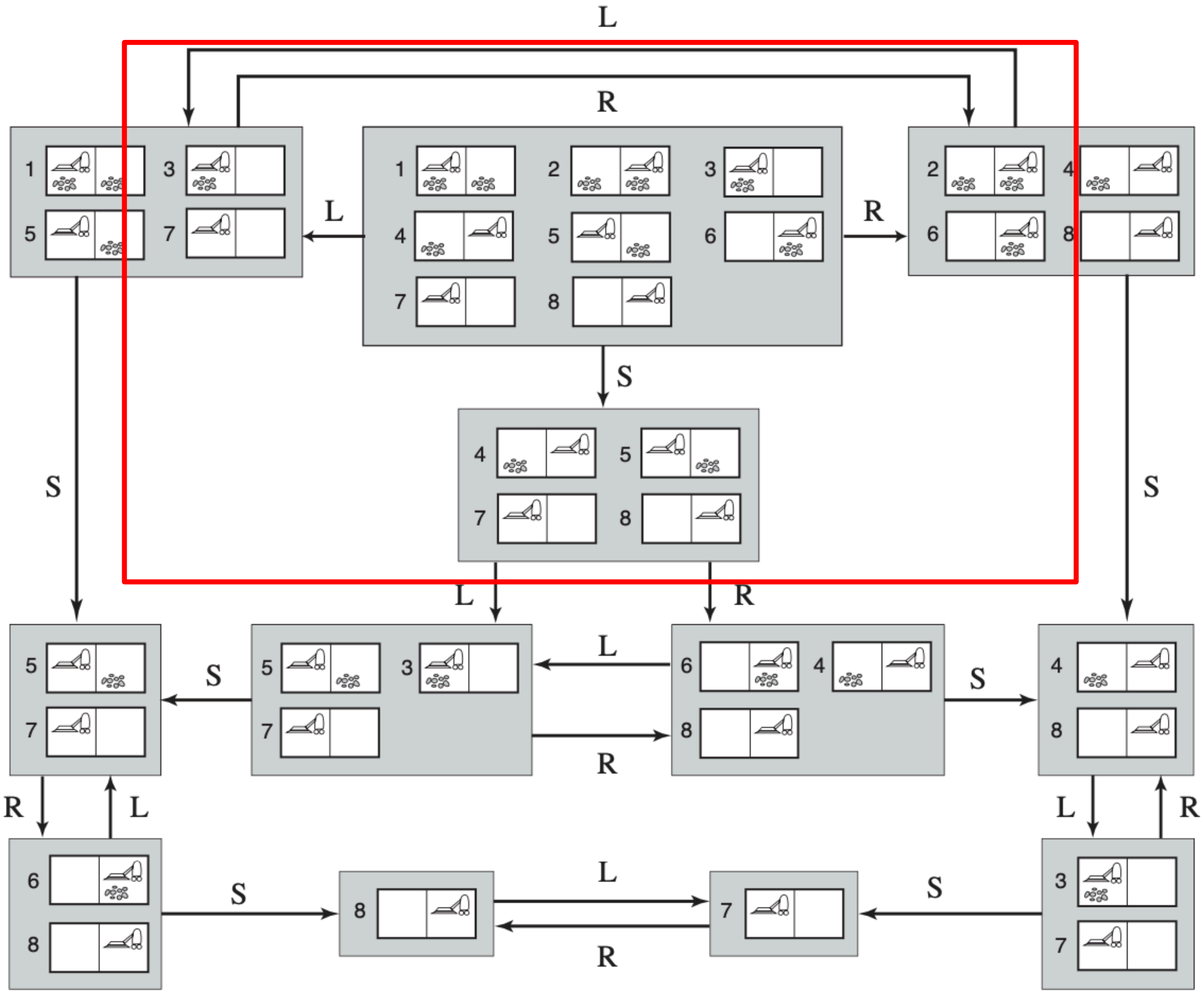
function OR-SEARCH(*state*, *problem*, *path*) **returns** *a conditional plan, or failure*
  if *problem*.GOAL-TEST(*state*) **then return** the empty plan
  if *state* is on *path* **then return** *loop*
  $cyclic - plan \leftarrow None$
  for each *action* **in** *problem*.ACTIONS(*state*) **do**
    $plan \leftarrow$ AND-SEARCH(RESULTS(*state*, *action*), *problem*, [*state* | *path*])
    if $plan \neq failure$ **then**
      if *plan* is acyclic **then return** [*action* | *plan*]
      $cyclic - plan \leftarrow [action \mid plan]$
  if $cyclic - plan \neq None$ **then return** $cyclic - plan$
  return *failure*

---

function AND-SEARCH(*states*, *problem*, *path*) **returns** *a conditional plan, or failure*
  $loopy \leftarrow True$
  for each s_i **in** *states* **do**
    $plan_i \leftarrow$ OR-SEARCH(s_i, *problem*, *path*)
    if $plan_i = failure$ **then return** *failure*
    if $plan_i \neq loop$ **then** $loopy \leftarrow False$
  if not *loopy* **then**
    return [**if** s_1 **then** $plan_1$ **else if** s_2 **then** $plan_2$ **else** ... **if** s_{n-1} **then** $plan_{n-1}$ **else** $plan_n$]
  return *failure*

4.7

4.7 In Section 4.4.1 we introduced belief states to solve sensorless search problems. A sequence of actions solves a sensorless problem if it maps every physical state in the initial belief state b to a goal state. Suppose the agent knows $h^*(s)$, the true optimal cost of solving the physical state s in the fully observable problem, for every state s in b. Find an admissible heuristic $h(b)$ for the sensorless problem in terms of these costs, and prove its admissibilty. Comment on the accuracy of this heuristic on the sensorless vacuum problem of Figure 4.14. How well does A* perform?



L
R
1 3
5 7
L
1 2 3
4 5 6
7 8
R
2 4
6 8
S
S
4 5
7 8
S
L
R
5
7
S
5 3
7
L
R
6 4
8
S
4
8
R
L
L
R
6
8
S
8
L
R
7
S
3
7

4.10

4.10 Consider the sensorless version of the erratic vacuum world. Draw the belief-state space reachable from the initial belief state $\{1, 2, 3, 4, 5, 6, 7, 8\}$, and explain why the problem is unsolvable.

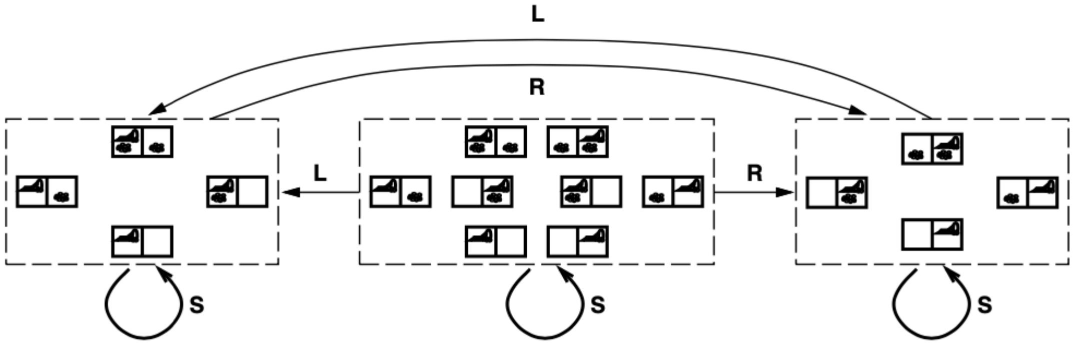


Figure S4.3 The belief state space for the sensorless vacuum world under Murphy's law.

4.12

4.12 Suppose that an agent is in a 3×3 maze environment like the one shown in Figure 4.19. The agent knows that its initial location is (1,1), that the goal is at (3,3), and that the actions *Up*, *Down*, *Left*, *Right* have their usual effects unless blocked by a wall. The agent does *not* know where the internal walls are. In any given state, the agent perceives the set of legal actions; it can also tell whether the state is one it has visited before.

a. Explain how this online search problem can be viewed as an offline search in belief-state space, where the initial belief state includes all possible environment configurations. How large is the initial belief state? How large is the space of belief states?

b. How many distinct percepts are possible in the initial state?

c. Describe the first few branches of a contingency plan for this problem. How large (roughly) is the complete plan?

Notice that this contingency plan is a solution for *every possible environment* fitting the given description. Therefore, interleaving of search and execution is not strictly necessary even in unknown environments.

Hint

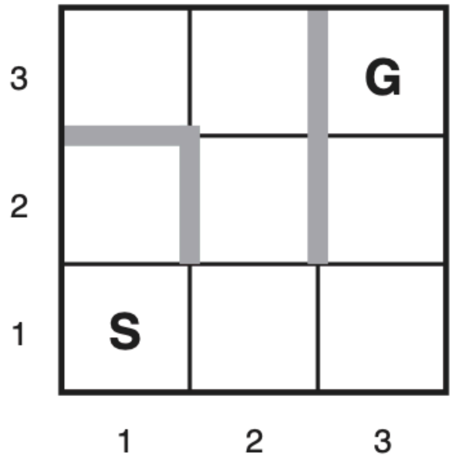
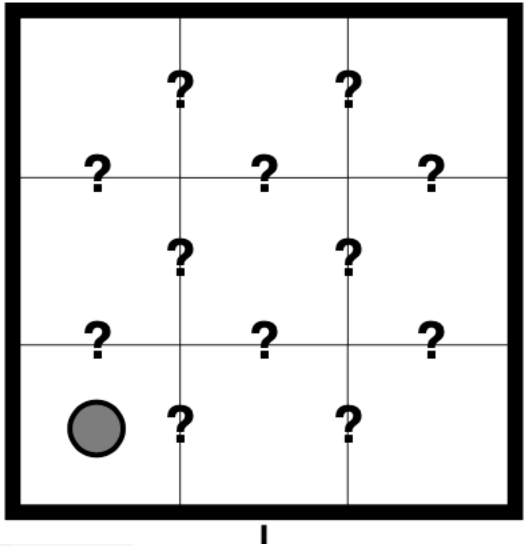


Figure 4.19 A simple maze problem. The agent starts at S and must reach G but knows nothing of the environment.

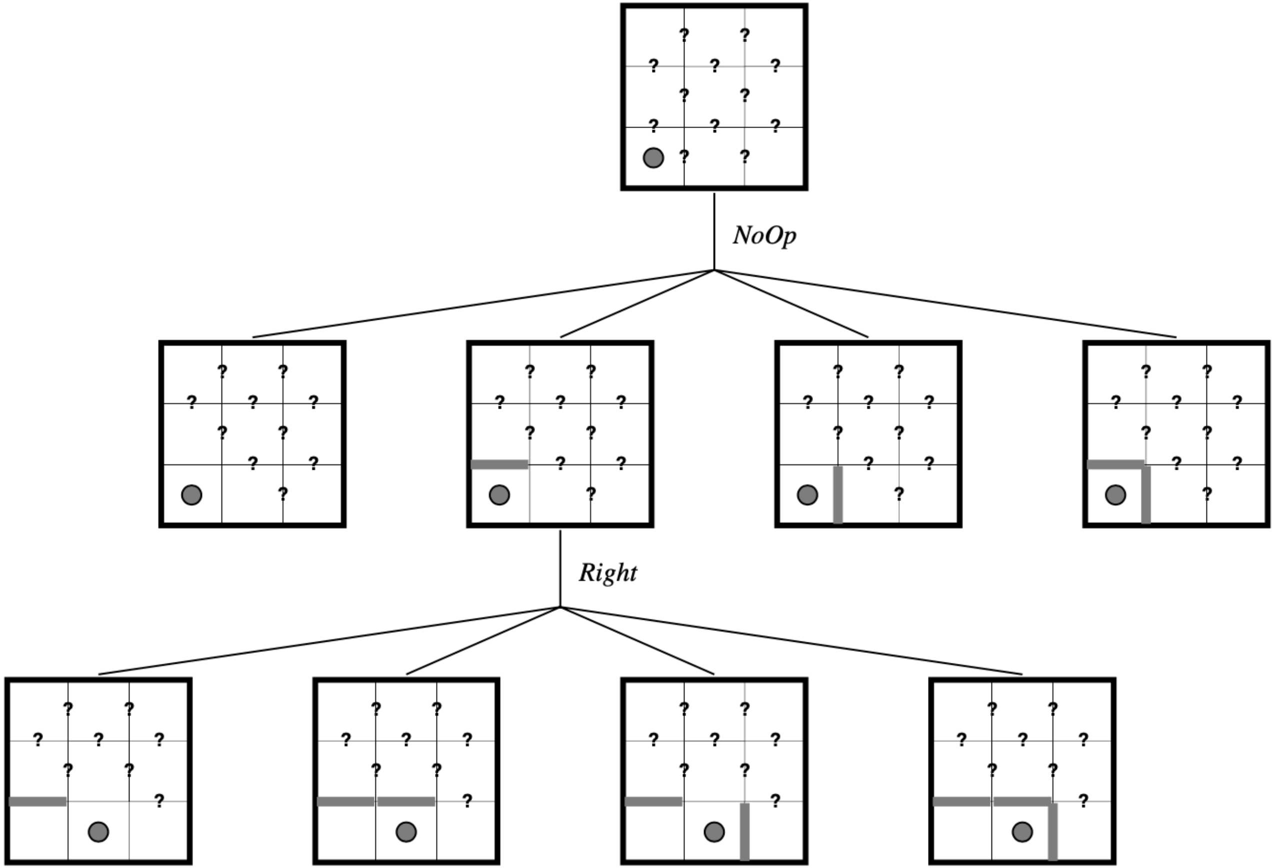
Hint



AND-OR search 2^{10}, 2^{12}

Distinct percepts 2^2, 2^4, 2^8

Hint



4.14

4.14 Like DFS, online DFS is incomplete for reversible state spaces with infinite paths. For example, suppose that states are points on the infinite two-dimensional grid and actions are unit vectors $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$, tried in that order. Show that online DFS starting at $(0, 0)$ will not reach $(1, -1)$. Suppose the agent can observe, in addition to its current state, all successor states and the actions that would lead to them. Write an algorithm that is complete even for bidirected state spaces with infinite paths. What states does it visit in reaching $(1, -1)$?

Hint

backtrack from to a previous state

Iterative deepening search(IDS)

- Depth1:
- Depth2:
- Depth3:

⋮

Due: 2020-12/15

作業名稱：學號 _ 姓名 _ HW2

E-mail：jerrytsai860216@gmail.com

Thank you