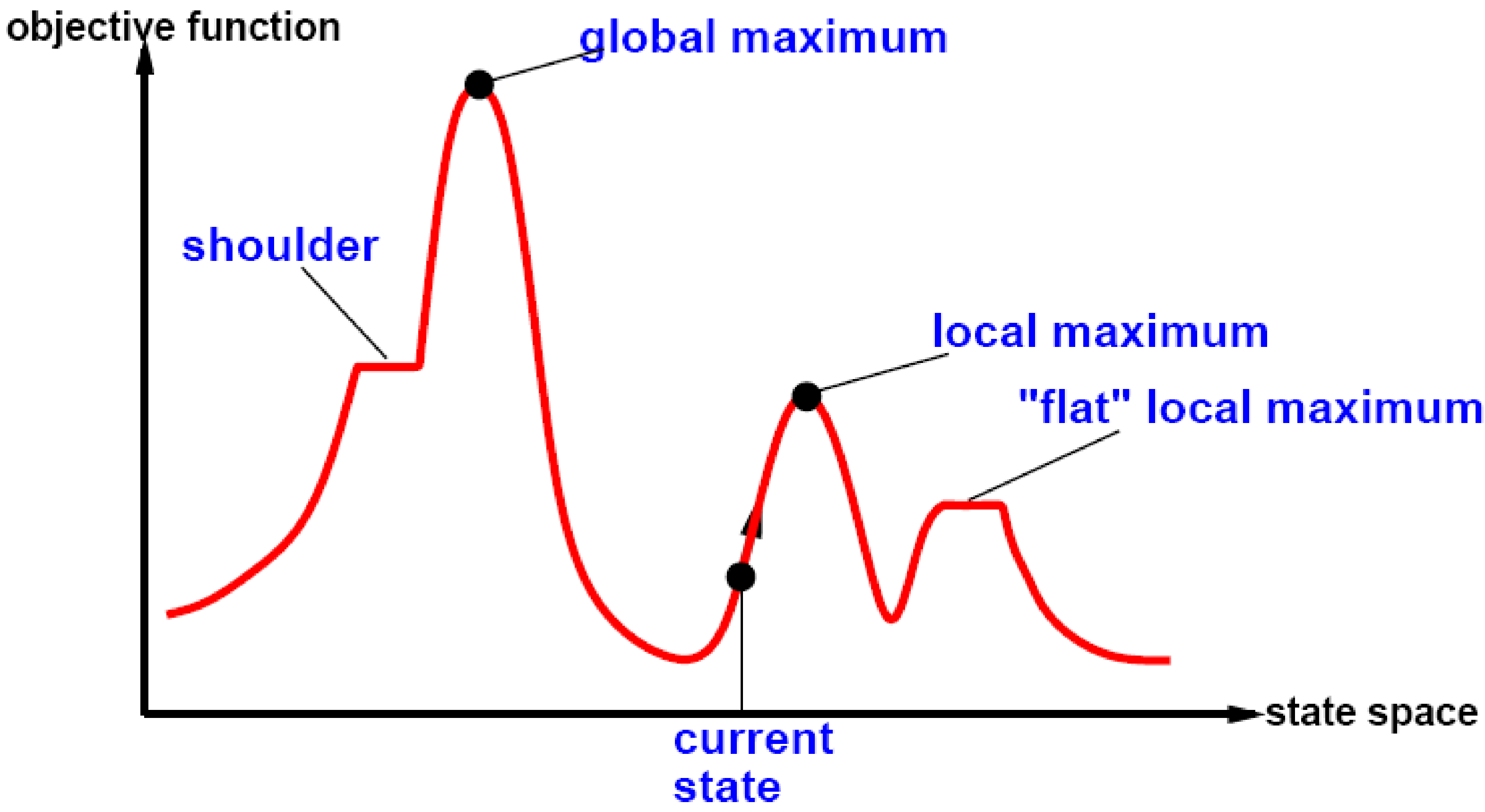
Local Search

Local Search and Optimization

- Optimization: To find a state that optimizes (minimizes or maximizes) an **objective function**.
- Here (sections 4.1-4.2 in the textbook) we use searching for the purpose of optimization.
- CSPs (e.g., 8-queens) can be solved using an objective function that represents the degree of satisfaction (or conflict) to the constraints.
- Exploitation-vs-Exploration is still an important issue in local search methods.

State-Space Landscape



- An example of an objective function to be maximized.
- The objective function is a function of the states.
- It is useful to think of the objective function as a surface.

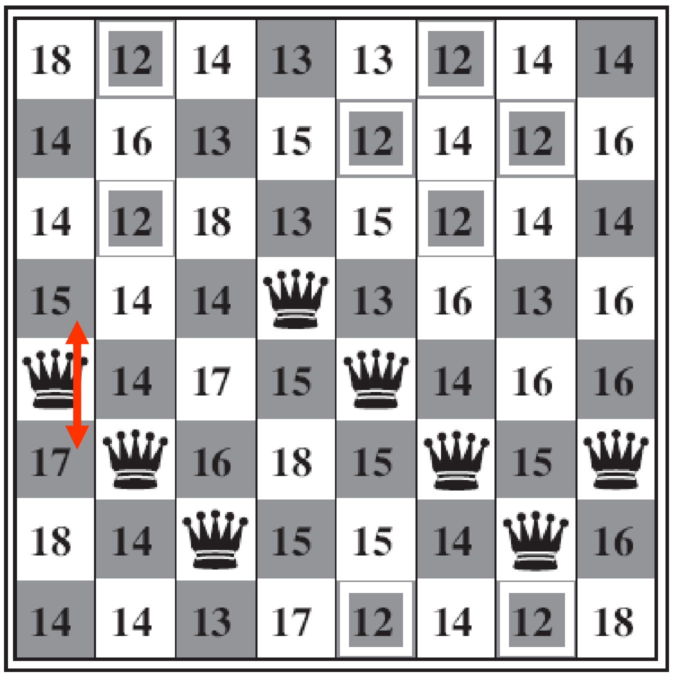
Hill-Climbing Search

- Also called "greedy local search".
- From the current state, compute the objective function for all of its immediate successors.
- Always move to the successor state that optimizes the objective function (steepest ascend / descend).
- Simple and efficient.
- Easy implementation: no queue, no tree, …
- Can get stuck at suboptimal states:
  - Local maximum / minimum
  - Ridge
  - Plateau (flat regions)

Hill-Climbing Search: Example

An illustration using the 8-queens problem:

- A state has 8 queens, one on each column.
- Each action can move a queen within its column.



Which queen to move next, and to which cell?

- Success rate: 14% (average ≈ 4 steps).
- Probability of getting stuck: 86% (average ≈ 3 steps).

Variants of Hill-Climbing Search

To improve the success rate or efficiency:

- Stochastic hill climbing: Choose probabilistically among several moves)
- First-choice hill climbing: Choose the first move found to be better than the current state (when it is too costly to evaluate all successors)
- Random sideway moves: Allows moves to states with the same objective function
  - Escape from shoulders
  - Need to avoid infinite loops (can limit the number of allowed consecutive sideway moves)
- Random restart

Simulated Annealing

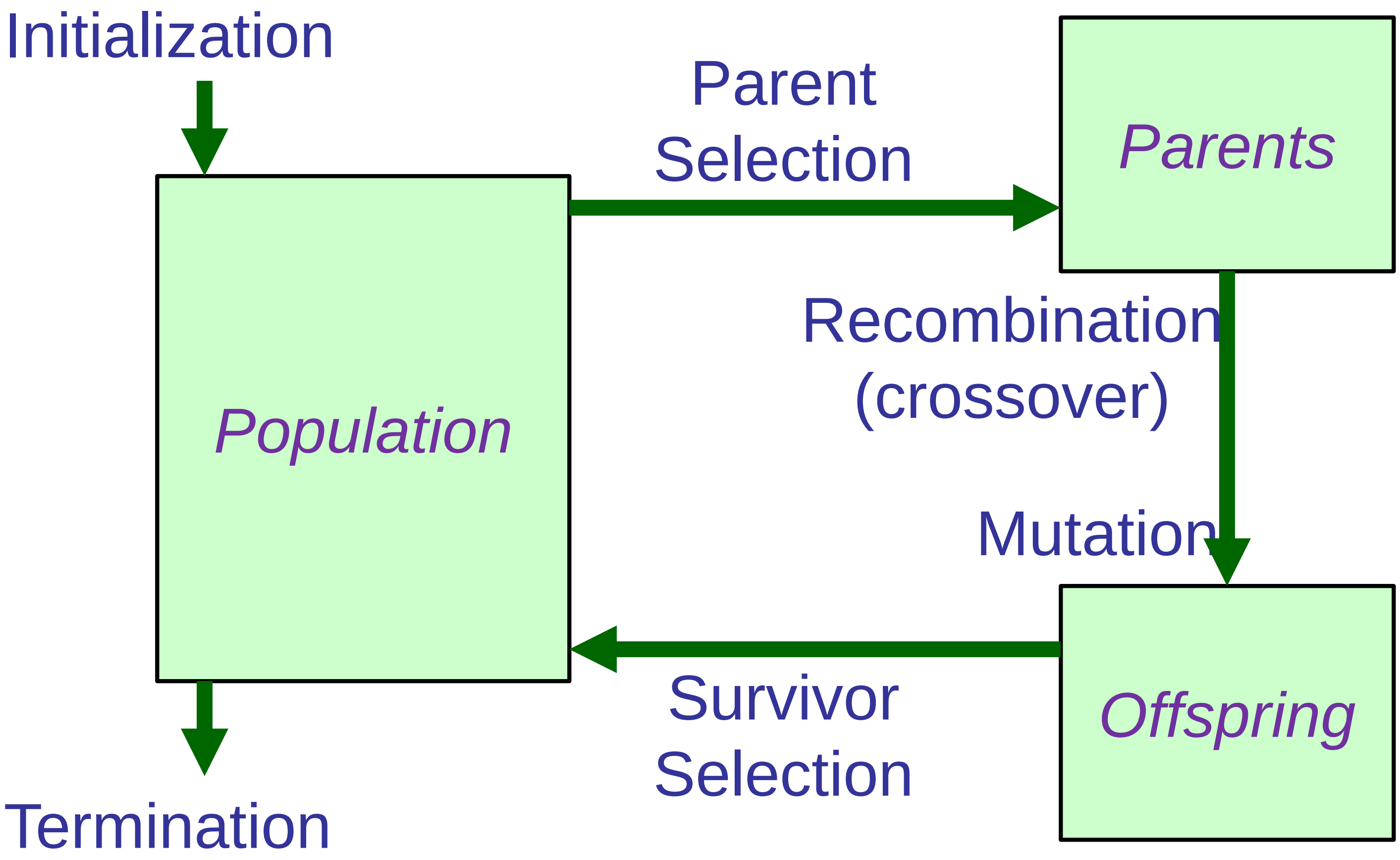
Difference from hill-climbing:

- Rules of selecting a successor:
  - Randomly pick a successor
  - If it is a better one, select it
  - Otherwise, still select it with a probability
- The probability of selecting a worse state is regulated by a variable T (temperature); higher T allows larger probability.
- This leads to opportunities for escaping from local optimums.
- Temperature is gradually reduced to zero (to facilitate convergence).

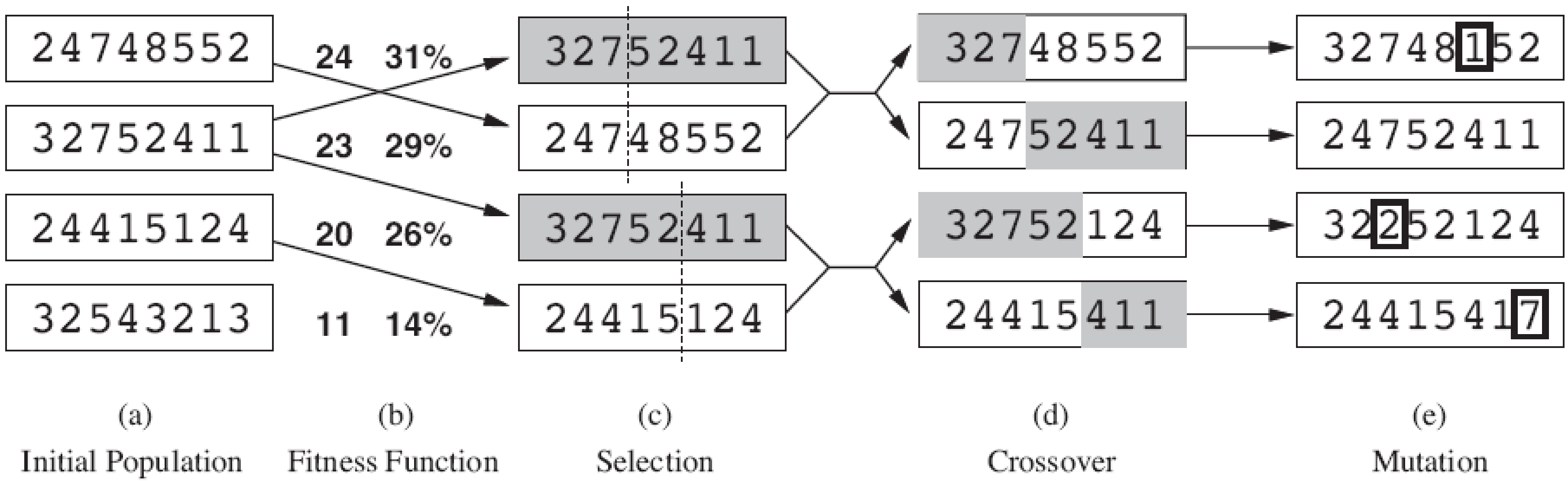
Genetic Algorithms

- A state is represented as a string. (e.g., the vertical positions of the 8 queens). This is the gene of the state.
- Multiple simultaneous searches by a **population** of individuals (each individual having its own state).
- **Fitness function**: For evaluating the goodness of an individual.
- Each step of the search is a **generation**. Things that can happen in each generation:
  - **Crossover** (two individuals exchange part of their genes)
  - **Mutation** (local move of an individual with a small probability)
  - Selection: Removal of worse individuals and addition of newly generated ones.

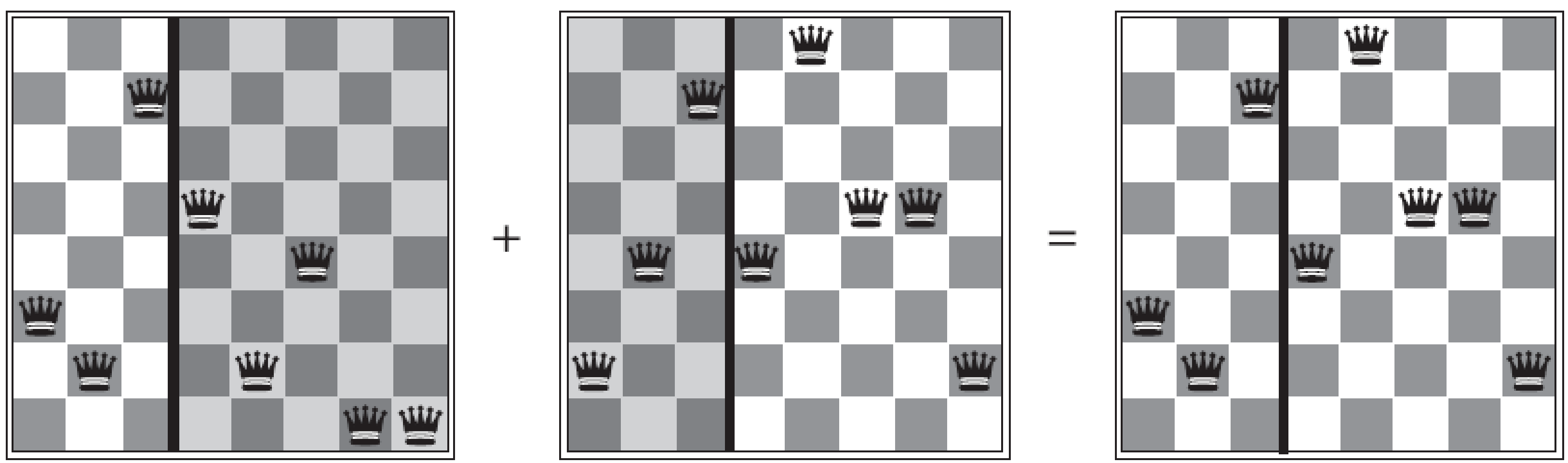
How Genetic Algorithms Work



Genetic Algorithms: Example



An illustration of crossover:



Local Search in Continuous Spaces

- Some ways to handle a continuous world:
  - Discretize it!
  - Steepest ascend / descent (a form hill-climbing).
  - Find local maximums/minimums analytically or numerically by setting the gradient of the objective function to zero. (This is not searching.)

Meta-heuristics

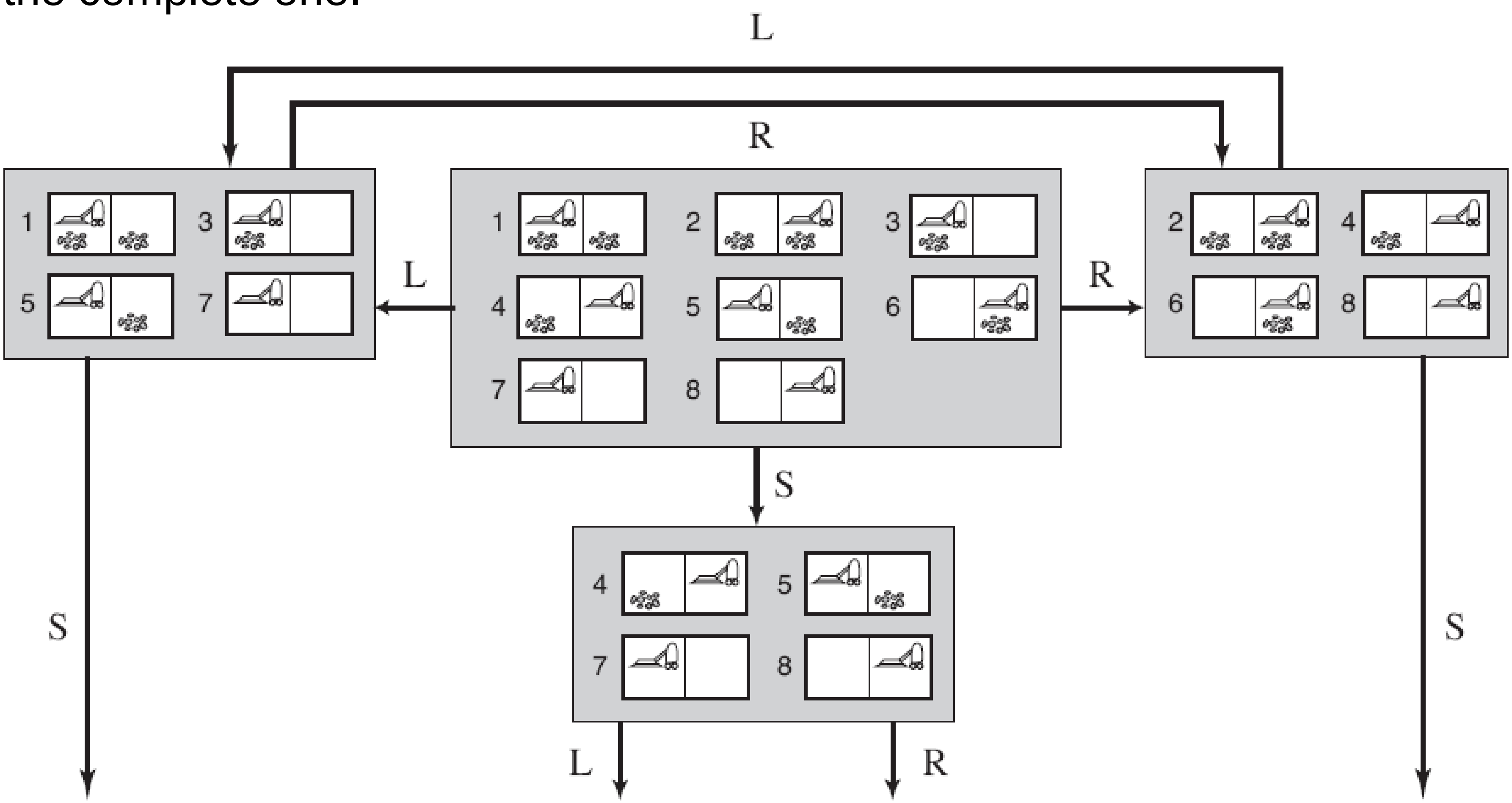
- **Meta-heuristics** are heuristic methods (not functions) for optimization to improve the efficiency and/or likelihood of finding good solutions.
- They are designed to be problem-independent.
- Simulated annealing and genetic algorithms are two representative and widely used meta-heuristics.
- Many are nature-inspired.
- Some other important ones:
  - Particle swarm optimization (PSO)
  - Ant colony optimization (ACO)
  - Artificial immune systems
  - …

Searching with No Observations

- Known: The state space.
- Unknown: The current state.
- Sensorless (comformant) problem: The agent has no percept to determine the state.
  - **Belief state**: Each belief state contains the possible real states.
  - An action from a belief state results in a new belief state containing the real states that can result from applying the action to any of the real states in the original belief state.
  - Solution can be found by searching in the belief-state space. (A goal state here is one that contains only goal states in the underlying state space.)

Searching with No Observations

Partial belief state space of the vacuum-cleaner world. See the textbook for the complete one.



A solution: [*Right*, *Suck*, *Left*, *Suck*]

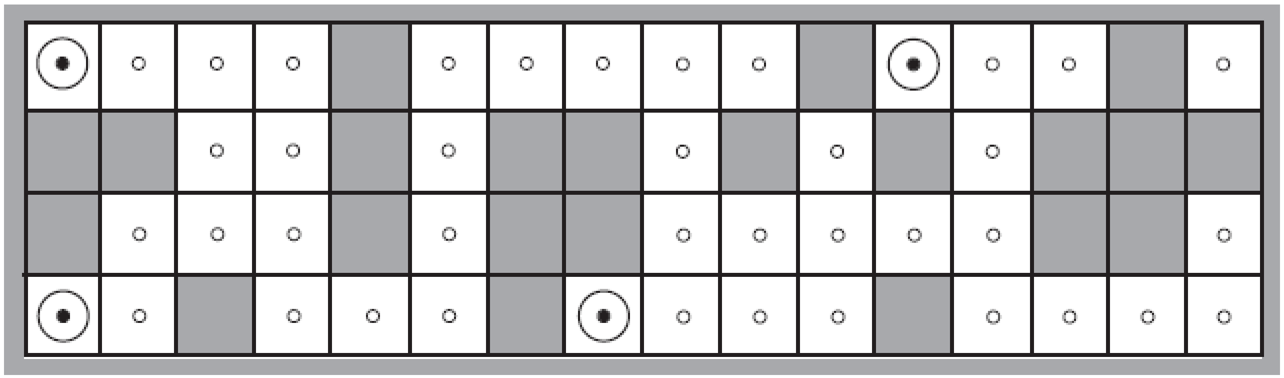
Searching with Partial Observations

- Some percept is available.
- The percept received after an action further limits the possible states.
- The solution is a conditional plan (depending on the updated belief state) instead of a sequence of actions.
- Example for the local-sensing vacuum-cleaner world:
  - The percept includes the location and local dirt condition.
  - Initial percept is [A, $Dirty$].
  - Solution: [$Suck$, $Right$, **if** $Bstate = \{6\}$ **then** $Suck$]

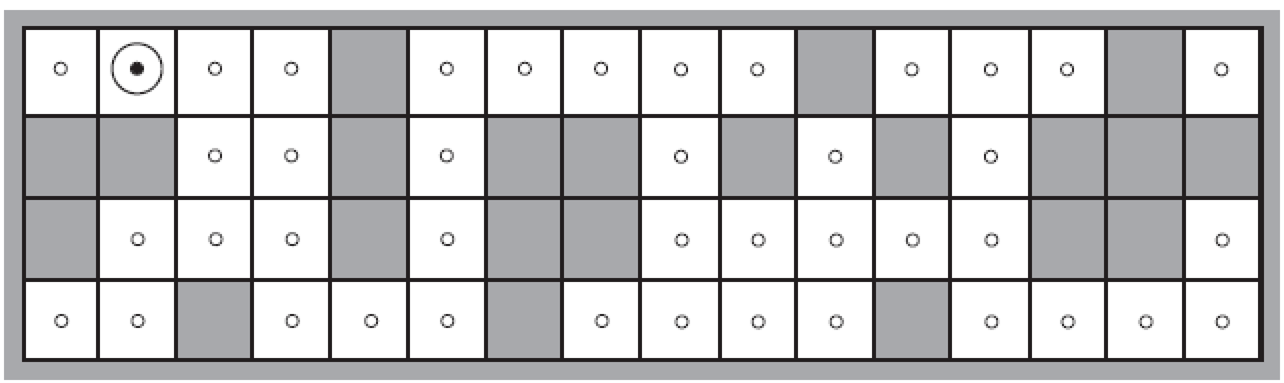
Here the solution only involves belief states, not actual states.

Searching with Partial Observations

- Example: Localization with local sensing (map is known).
- Percept: Whether there are obstacles in the four (E,W,N,S) directions.
- Action/percept sequence: *NSW* → *Right* → *NS*



(a) Possible locations of robot after $E_1 = NSW$



(b) Possible locations of robot After $E_1 = NSW, E_2 = NS$