CMSC 671
Fall 2010

Tue 9/14/10

Local Search and Optimization Problems

Prof. Laura Zavala, laura.zavala@umbc.edu, ITE 373, 410-455-8775

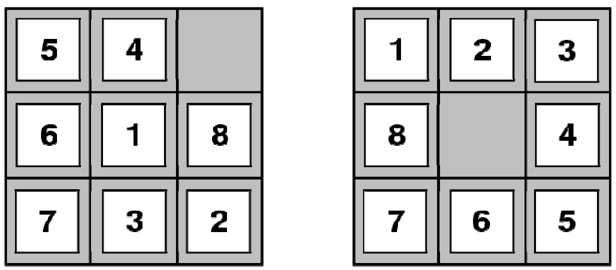
Saving the path to the goal (2)

- Informed and Uninformed search methods we have seen so far
  - Breadth-first, uniform-cost, depth-first, depth-limited, iterative deepening, bidirectional
  - Greedy, A*, IDA*, SMA*
- The path to the goal is available as part of the solution

Saving the path to the goal

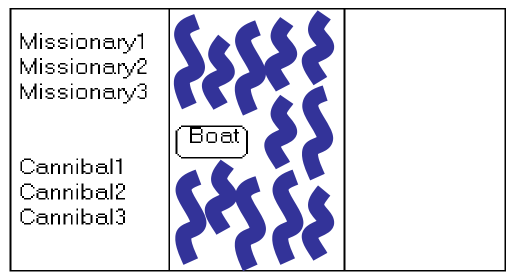
- In many problems the path to the goal is **irrelevant**
- Can you tell which ones?

8-Puzzle

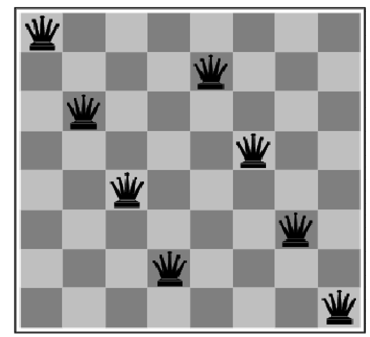


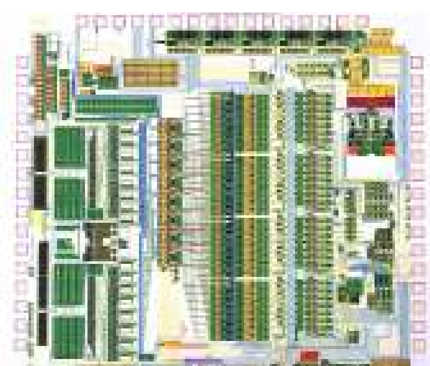
Start State Goal State

Missionaries & Cannibals



N-Queens





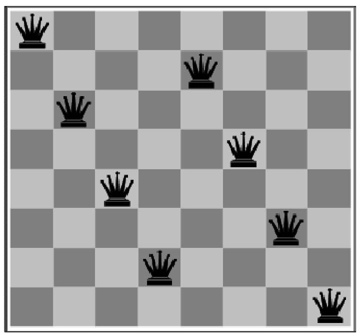
Integrated Circuit Design

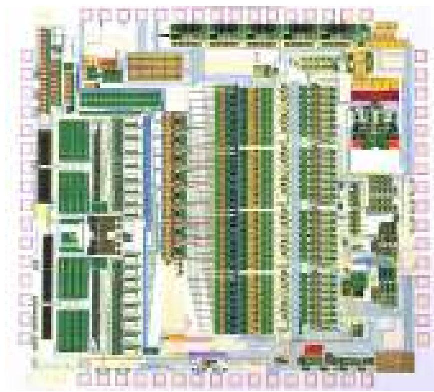
Water Jug Problem

Saving the path to the goal (3)

- In many problems the path to the goal is **irrelevant**

N-Queens





Integrated Circuit Design

- Vehicle routing, job-shop schedulling, process scheduling, etc.
- In general: **Optimization Problems**

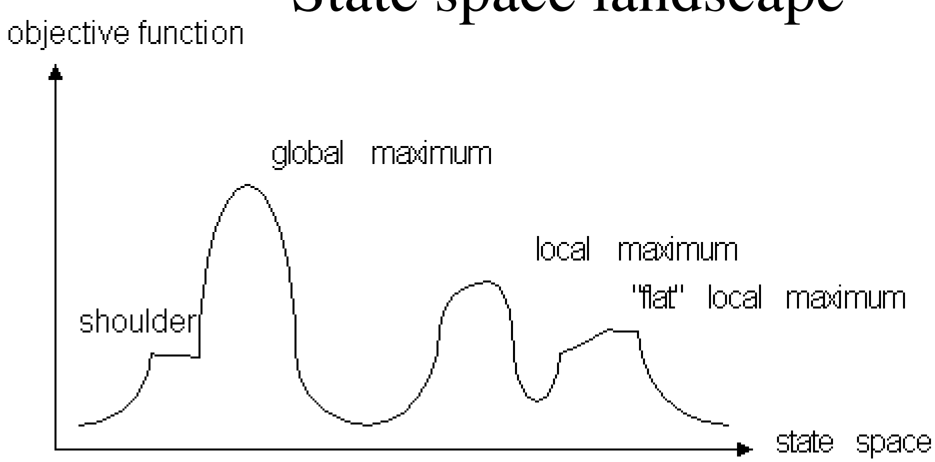
Optimization problem

- The aim is to find the best state according to an **objective function**.
- The objective function determines how good a solution is.

Local search

- Keep a single "current" state, try to improve it.
  - Very memory efficient (only remember current state)

State space landscape



Elevation - Objective function
Location – State
Depending on app., aim is either to find lowest valley or highest peak

Hill-climbing search

- If there exists a successor for the current state n such that
  - h(s) < h(n)
  - then move from n to s. Otherwise, halt at n.
- Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.
- Similar to Greedy search in that it uses h, but does not allow backtracking or jumping to an alternative path since it doesn't "remember" where it has been.

Exploring the Landscape

- **Local Maxima**: peaks that aren't the highest point in the space
- **Plateaus:** the space has a broad flat region that gives the search algorithm no direction (random walk)
- **Ridges:** sequence of local maxima very difficult to navigate

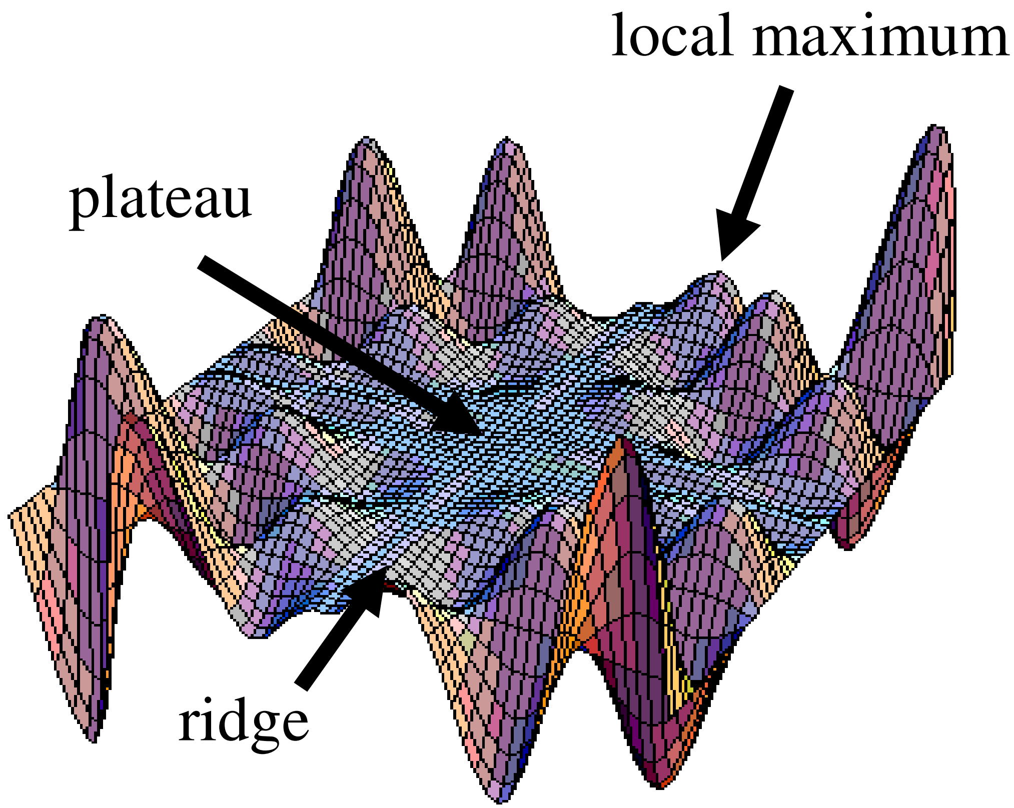
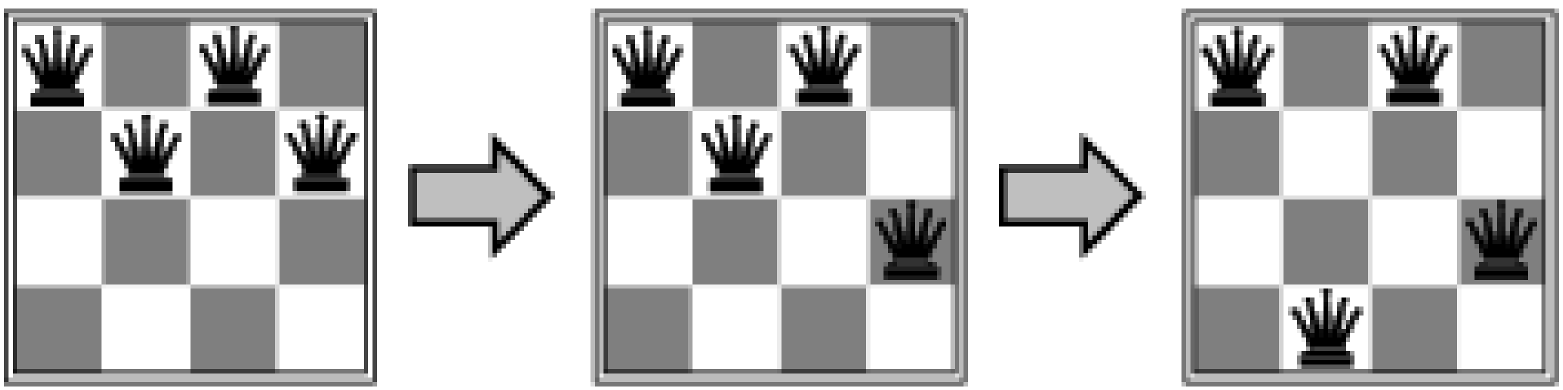


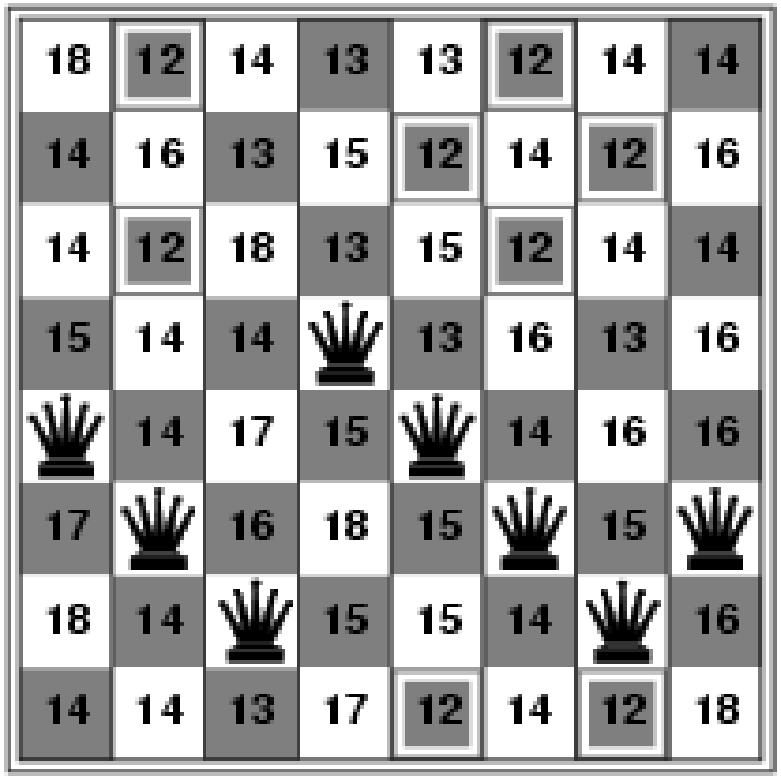
Image from: http://classes.yale.edu/fractals/CA/GA/Fitness/Fitness.html

Example: n-queens problem

- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal



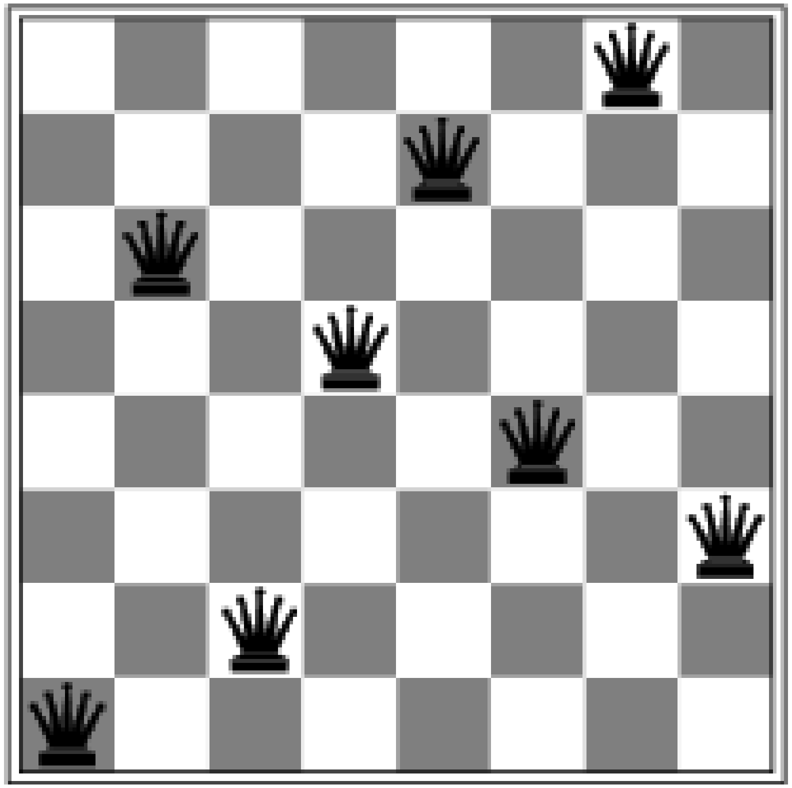
8-queens problem



Numbers indicate h if we move a queen in its corresponding column

- h = number of pairs of queens that are attacking each other, either directly or indirectly
- $h = 17$ for the given state

Local minima



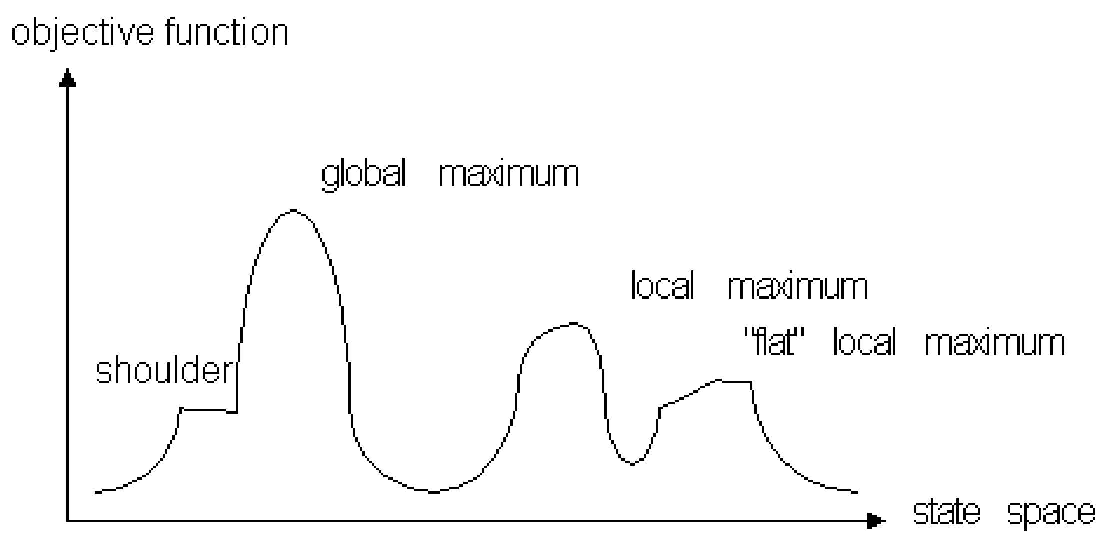
- Local minimum, $h = 1$

How do we get out of this local minima?

Hill-climbing search Local minima/maxima

- Depending on initial state, can get stuck in local minima (maxima)

State space landscape



Hill-climbing search issues

- Not complete since the search will terminate at "local minima (maxima)," "plateaus," and "ridges.“
- Some problem spaces are great for hill climbing and others are terrible.
  - Depends very much on the shape of the state-space landscape

Alternatives to hill climbing

- Problems: local maxima, plateaus, ridges
- Remedies:
  - **Random restart:** keep restarting the search from random locations until a goal is found.
  - **Stochastic Hill Climbing**: randomly choosing from among the uphill moves (prob. according to steepness)
  - **First choice hill climbing**: randomly generates successors, one by one, until a better one is found
    - Good when thousands of successors

Simulated annealing

- Simulated annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum-energy crystalline structure (the annealing process) and the search for a minimum [or maximum] in a more general system.
- We “shake” the surface to bounce out of a local minima.

Simulated annealing (2)

- SA can avoid becoming trapped at local minima
- SA uses a random search that accepts changes that increase objective function f, **as well as** some that **decrease** it (i.e. it accepts bad moves).
- SA uses a control parameter T, which by analogy with the original application is known as the system "**temperature**" (shaking intensity).
- The higher the temperature, the more likely it is that a bad move can be made.
- T starts out high and gradually decreases toward 0.
- Widely used in VLSI layout and airline scheduling

Simulated annealing (3)

- A “bad” move from A to B is accepted with a probability

$$P(\text{move}_{A\to B}) = e^{(f(B)-f(A))\ /T}$$

- The higher the temperature, the more likely it is that a bad move can be made.
- As T tends to zero, this probability tends to zero, and SA becomes more like hill climbing
- If T is lowered slowly enough, SA is complete and optimal.

The simulated annealing algorithm

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"
  static: current, a node
          next, a node
          T, a "temperature" controlling the probability of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]
      if T=0 then return current
      next ← a randomly selected successor of current
      ΔE ← VALUE[next] – VALUE[current]
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

Local beam search

- Begin with k random states
- Generate all successors of these states
- Keep the k best states

- Can suffer lack of diversity among k states (expensive version of hill climbing)
- Stochastic beam search: Probability of keeping a state is *a function* of its heuristic value

Genetic algorithms

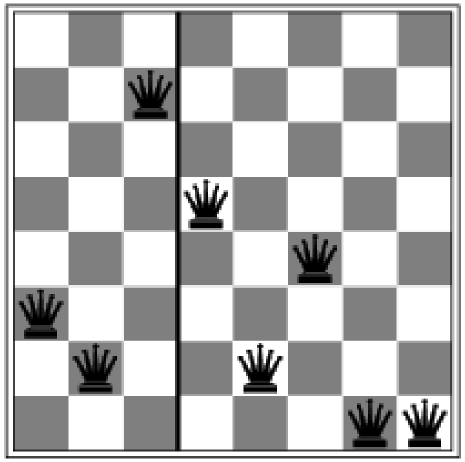
- Similar to stochastic beam search, but new states are generated by "reproducing" (combining via crossover) **two parent states** (selected according to their *fitness)* rather than modifying a single state.

Genetic algorithms (2)

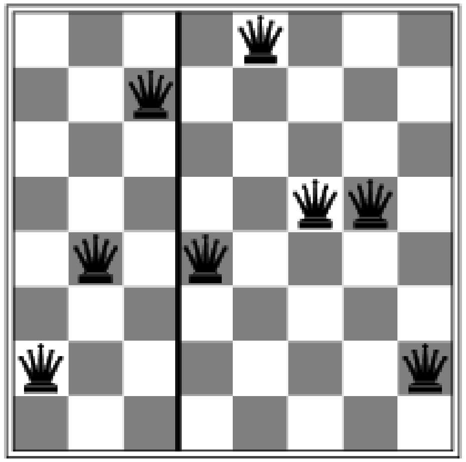
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s)
- Evaluation function (fitness function).
  - Higher values for better states.

1. Start with k randomly generated states (population)
2. Produce the next generation of states by
   1. selection
   2. crossover
   3. mutation

Genetic algorithms: Example



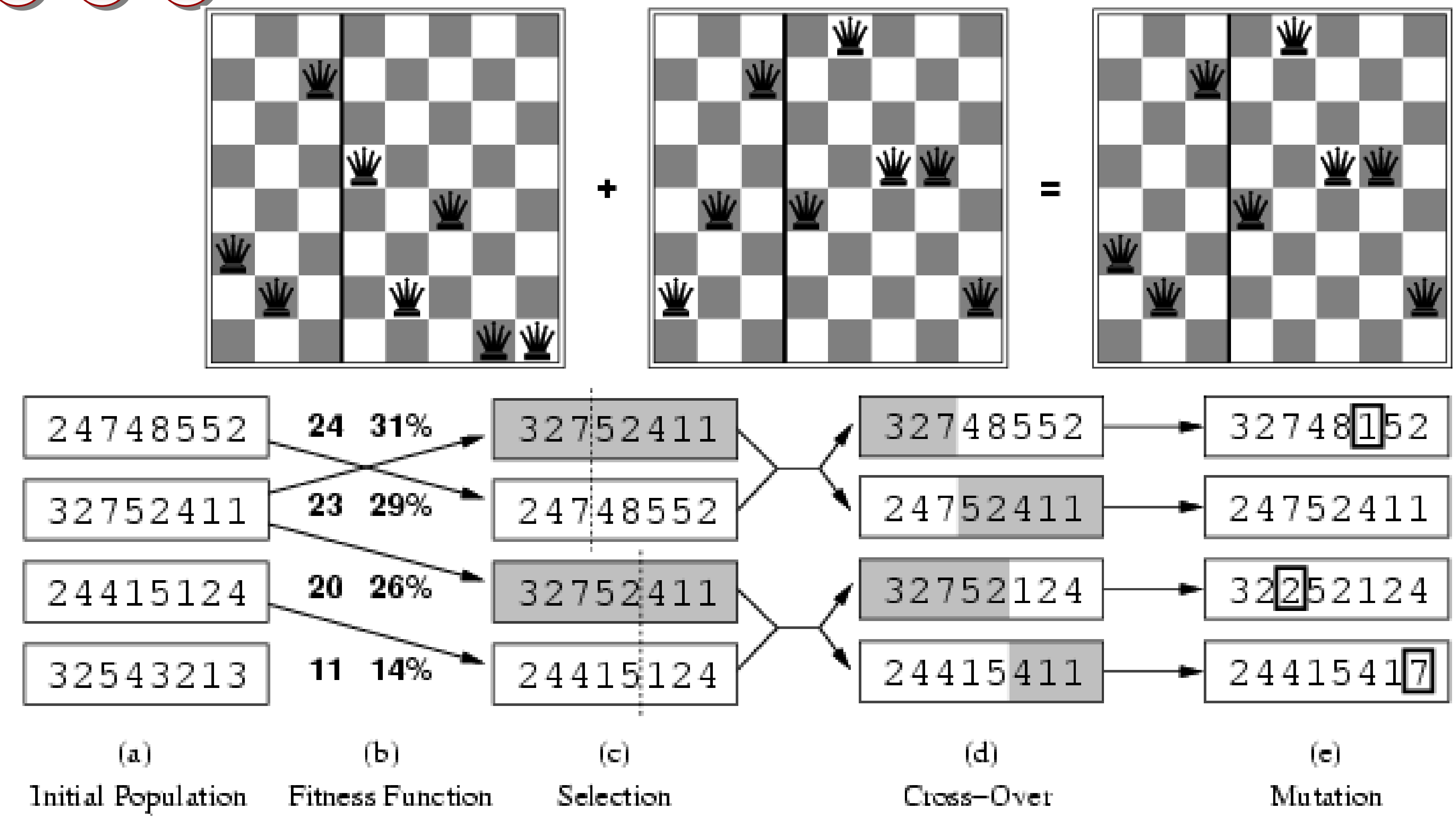
32752411
23



24748552
24

- Fitness function: number of non-attacking pairs of queens
  - min = 0, max = 8 × 7/2 = 28

Genetic algorithms: Example (2)



Fitness-based stochastic selection :

P(24748552) = 24/(24+23+20+11) = 31%

P(32752411) = 23/(24+23+20+11) = 29%

More on genetic algorithms

- Mimic the process of natural evolution
- Encoding used for the “genome” of an individual strongly affects the behavior of the search
- Genetic algorithms / genetic programming are a large and active area of research

Tabu search

- A simple local search but with a memory

- Problem: Hill climbing can get stuck on local maxima
- Solution: Maintain a list of *k* previously visited states, and prevent the search from revisiting them

Online search

- Interleave computation and action (search some, act some)
- Exploration: Can't infer outcomes of actions; must actually perform them to learn what will happen

- Competitive ratio = Path cost found* / Path cost that could be found**
  - * On average, or in an adversarial scenario (worst case)
  - ** If the agent knew the nature of the space, and could use offline search

- Relatively easy if actions are reversible (ONLINE-DFS-AGENT)
- LRTA* (Learning Real-Time A*): Update h(s) (in state table) based on experience
- More about online search and nondeterministic actions later un the course …

Summary: Local search

- **Hill-climbing algorithms** keep only a single state in memory, but can get stuck on local optima.
- **Simulated annealing** escapes local optima, and is complete and optimal given a "long enough" cooling schedule.
- **Genetic algorithms** can search a large space by modeling biological evolution.
- **Online search** algorithms are useful in state spaces with partial/no information.

Thanks for coming -- see you next Thursday!