Local Search

- Reading: Russell and Norvig, ch. 4.1-4.2
  - The material here is an edited version of Russell's slides
- Hill-climbing
- Simulated annealing
- Local beam search
- Genetic algorithms

Context for local search

- Optimization problems where path to goal state does not matter
- Each state includes all information for problem solution
  - Ex: for Traveling Salesman, state includes complete travel sequence
  - Goal is to find optimal solution
- Iterative improvement algorithms:
  - Start with a solution to the problem
  - Each iteration, improve solution
  - Constant space
    - What do we need to store?
  - Can use for both offline and online search

Example: Traveling Salesman

- Start with a complete tour of the cities
  - Goal is to find cheapest tour – money, energy, time are options for cost
- Each iteration, remove a pair of edges, try to find new pair of edges to reduce total cost.
- Ex: By Lady-shirakawa - created on gliffy, CC BY-SA 3.0, https://en.wikipedia.org/wiki/Travelling_salesman_problem#/media/File:Showing_a_step_of_the_two-opt_heuristic.png

Example: Traveling Salesman

- Variants of this approach can get within 1% of optimal solution very quickly, even for thousands of cities
  - From https://en.wikipedia.org/wiki/Travelling_salesman_problem
    - Variants remove more than 2 edges before coming up with new solution
    - One problem with only swapping 2 edges is reaching a local minimum
    - Removing more edges slows algorithm, but produces better solutions

Example: N-queens

- Place N queens on N x N chess board without any two queens attacking each other (same row/column/diagonal)
- Each state places all N queens on board
  - May still have violations
- Each iteration: move one queen to reduce number of conflicts
- Ex: https://docs.jboss.org/drools/release/6.0.0.CR5/optaplanner-docs/html/localSearch.html shows solution for 4 queens

Hill-climbing (or gradient ascent/descent)

- “Like climbing Everest in thick fog with amnesia”

```
function Hill-Climbing(problem) returns a state that is
a local maximum
inputs: problem
local variables: current, neighbor (nodes)
current ← Make-Node(Initial-State[problem])
loop do
   highest-valued determined by objective function
   neighbor← highest-valued successor of current
   if Value[neighbor] ≤ Value[current]
   then return State[current]
   current ← neighbor
end
```

Challenges for hill-climbing

- “thick fog”
  - Only compare current node to “neighbor” nodes, no lookahead
- “amnesia”
  - Basic algorithm does not check if revisiting previous state
- “Everest”
  - Local maxima in objective function – higher “values” than neighbors
  - Randomly restarting at different states can help
    - But may require many restarts
    - Problem may have many, many local maxima

Challenges for hill-climbing

- Other “features” of concern:
  - Flat features: objective function has same value for a neighborhood of nodes
    - “shoulder”: flat portion is not even a local maxima
      - random lateral move (beyond neighbors) can detect this
    - Flat local maxima: values are higher than nodes on either side of neighborhood
      - Lateral moves may not detect this
  - For > 2D objective functions, “saddles”: maximum along one axis, minimum along another axis
- Lathrop (UC-Irvine)’s notes point out importance of defining “neighbor” and objective function

Hill-climbing examples

- Newton-Raphson root-finding method
  - Given: function f(x), f is continuous (derivative is defined)
  - Goal: find x such that f(x) = 0
  - State: value of x
  - Neighbor of state: $x_{k+1} = x_k - f(x_k) / f'(x_k)$
    - Algorithm follows tangent to x-axis to get next guess
  - Objective function: |f(x)|, where 0 indicates we've reached a goal state
  - Example of stationary point: f(x) = sin(x), current guess = $\pi/2$
    - Derivative is 0, so no neighbor
  - Example of unstable behavior: $f(x) = x^2 + 4$

Hill-climbing examples

- Lathrop has example for 8-queens:
  - https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/cs-171-05-LocalSearch.pdf slides 12,13

Simulated annealing

- In metallurgy, annealing is a process to make a metal easier to work with – see https://en.wikipedia.org/wiki/Annealing_(metallurgy)
  - Metal is heated to a desired temperature and then gradually cooled
  - It becomes easier to stretch and less hard
  - Important in semiconductor industry to allow depositing within silicon lattice of other elements
    - Changes electrical properties to allow creating transistors on silicon wafer

Simulated annealing

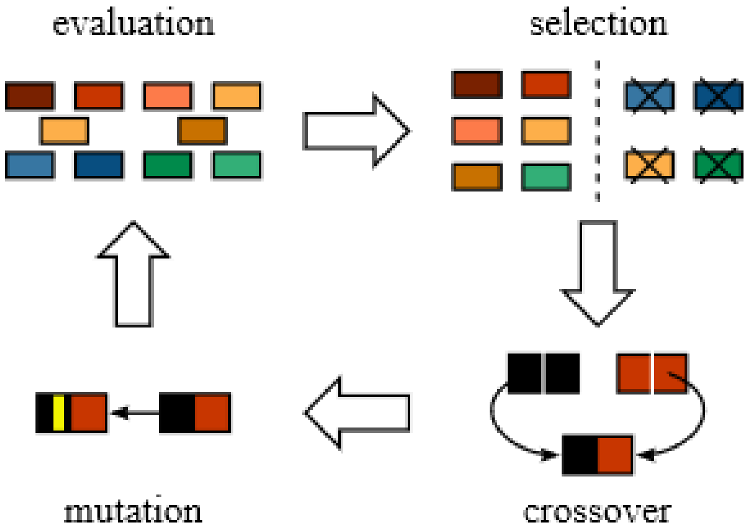
- Simulated annealing treats problem solving in a similar way
  - “Temperature” = time in text
    - Like material in annealing reaches final state when completely cooled, system decides on best node found at end of chosen time
  - Allows worse-looking nodes to be expanded, but as search continues, reduces probability of expanding worse-looking node
    - Still uses an evaluation function
    - The worse the node, the less likely it will be expanded
  - Motivation: allows escaping local maxima
- Applications: VLSI layout, airline scheduling

Local beam search

- Maintain top *k* nodes
  - Each iteration, choose top *k* from the successor nodes of all *k* nodes
- Problem: ending up with all nodes on same “hill”
  - Choose successors more randomly, though biased towards better nodes

Genetic algorithms

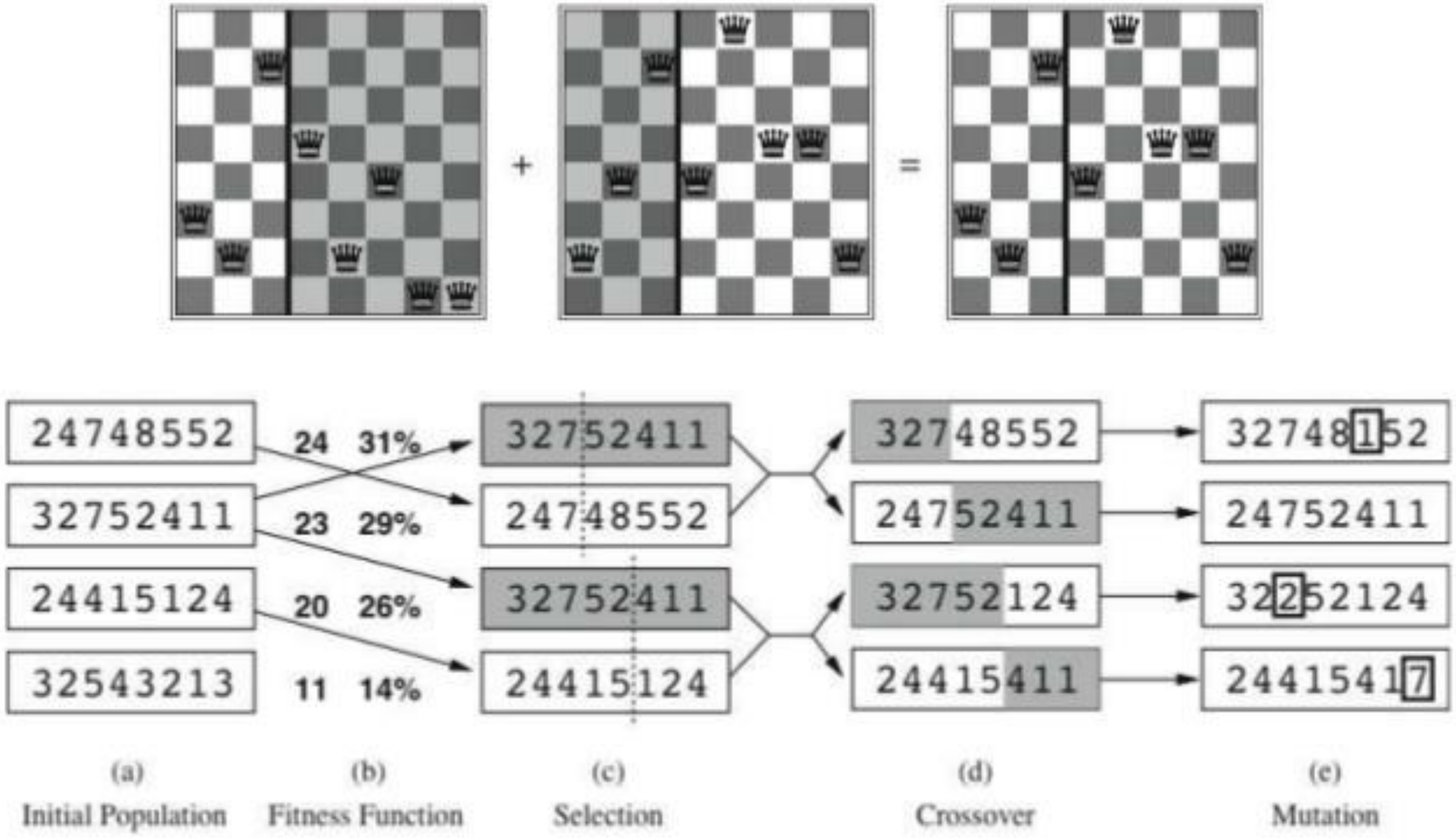
- Try to crudely mimic evolution in search process
- Need *fitness* evaluation function to select states
- *Crossover* : from pair of states with higher fitness, swap pieces
- *Mutation* : modify part of one state to generate successor



Genetic algorithms

- Implement by representing states as strings
  - Crossover = merging substrings from pair of states
  - Helpful if substrings are meaningful on their own
- Various termination criteria (https://en.wikipedia.org/wiki/Genetic_algorithm#Termination)
  - Solution found that satisfies specified criteria
  - Evolve for fixed number of generations
  - Run while enough resources (time, money)
  - Detect that local maximum has been reached
  - User-guided
  - Combination of above

Genetic algorithm: 8-queens example



Genetic algorithms

- Some issues (Wikipedia)
  - Why should this work?
  - Complexity of modeling
    - Number of features that can be modified
    - Time to compute fitness function
  - Convergence
    - May end up at local maximum, or even in just random part of state space
    - Unclear stopping criteria