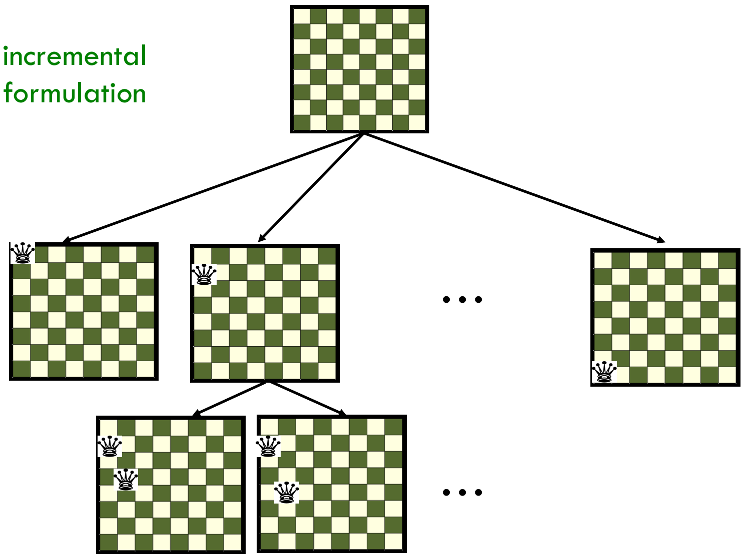
LOCAL SEARCH

Today

- Reading
  - Read AIMA Ch. 4.1-4.2
- Objectives
  - Simulated Annealing
  - Genetic algorithms
  - Gradient ascent

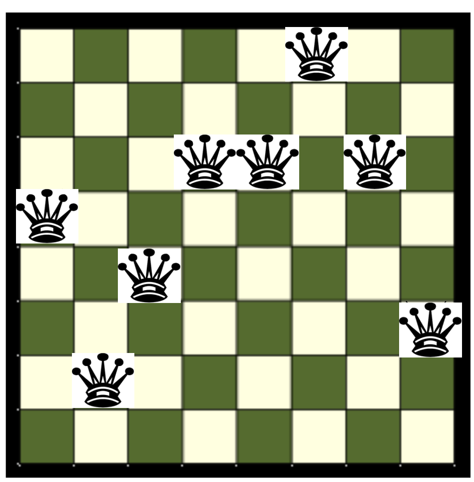
Recall the N-Queens problem

incremental formulation

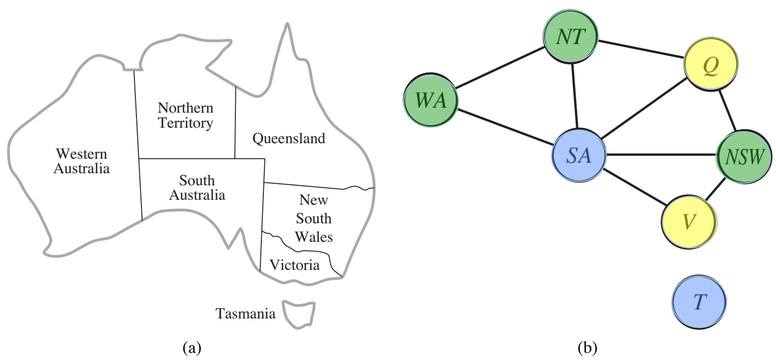


N-Queens alternative approach

complete state formulation



Map Colorings



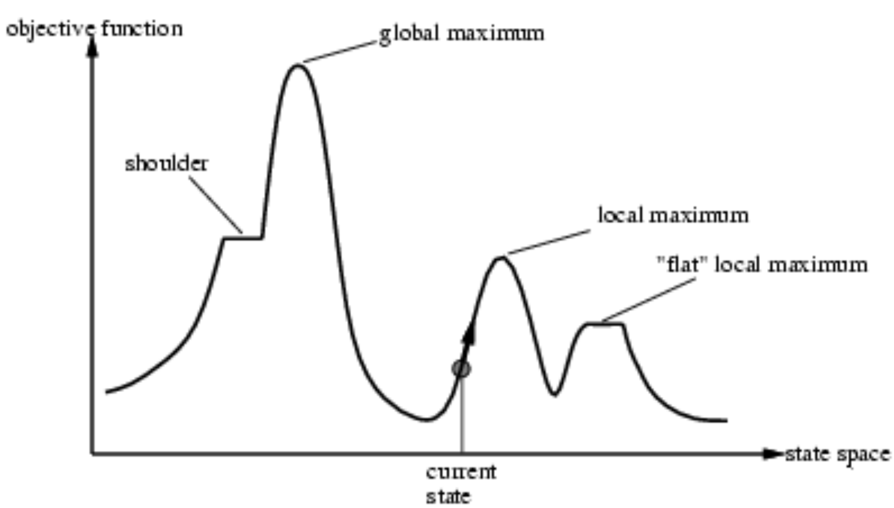
Local Search Algorithms

- The basic idea:
  1. Randomly initialize state
  2. If not goal state,
     a. make local modification to generate neighbor state OR
     b. enumerate all neighbor states and choose the best
  3. Repeat step 2 until goal state is found (or out of time)
- Requires the ability to *quickly*:
  - Generate a random (probably-not-optimal) state
  - Evaluate the quality of a state
  - Move to other states (well-defined neighborhood function)

Local Search Algorithms

- Use when path to goal is irrelevant
- Minimal memory requirements – keep track of current state only
- Algorithms include:
  - Hill-climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
  - Gradient descent (Newton-Rhapson)

Local Search Algorithms



Simulated Annealing

- Proposed in 1983
- Borrows from the idea of annealing in physics
  - Liquid cools too quickly, atoms solidify into sub-optimal configuration (local minimum)
  - Liquid cools slowly, atoms solidify into minimum energy configuration (global minimum)
- Early applications
  - Design of electronic systems
  - Traveling salesman

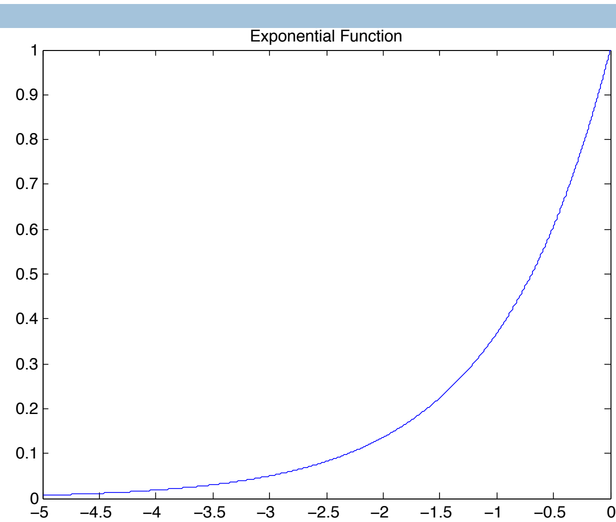
Simulated Annealing

- Determine a schedule that maps from the iteration (t) to the temperature (T)
- Propose a next state
  - If VALUE(next) > VALUE(curr), accept always
  - If VALUE(next) < VALUE(curr), accept with probability

$$e^{(VALUE(next)-VALUE(current))/T}$$

Assumes we're maximizing function

Simulated Annealing



Simulated Annealing

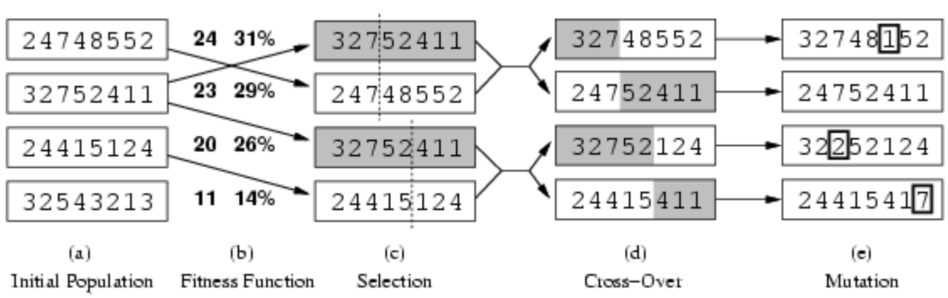
function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"
  local variables: *current*, a node
    next, a node
    T, a "temperature" controlling prob. of downward steps

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

Genetic Algorithms

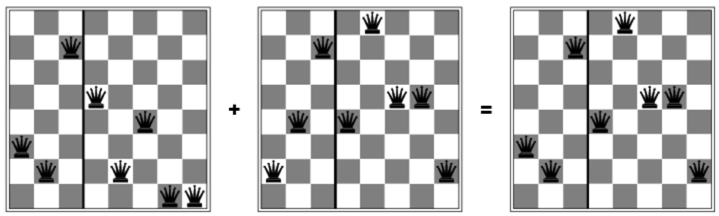
- Survival of the fittest!
  - Initialize population to N randomly generated states represented as strings over finite alphabet
  - Evaluate the fitness of each individual via fitness function (higher=better)
  - To generate a new population, repeat:
    - Randomly select 2 individuals proportional to their fitness
    - Randomly pick a crossover point and produce 2 new children
    - Randomly mutate each location of the new children

Genetic Algorithms



- Fitness function: number of non-attacking pairs of queens
- 24/(24+23+20+11) = 31%
- 23/(24+23+20+11) = 29% etc

Genetic Algorithms

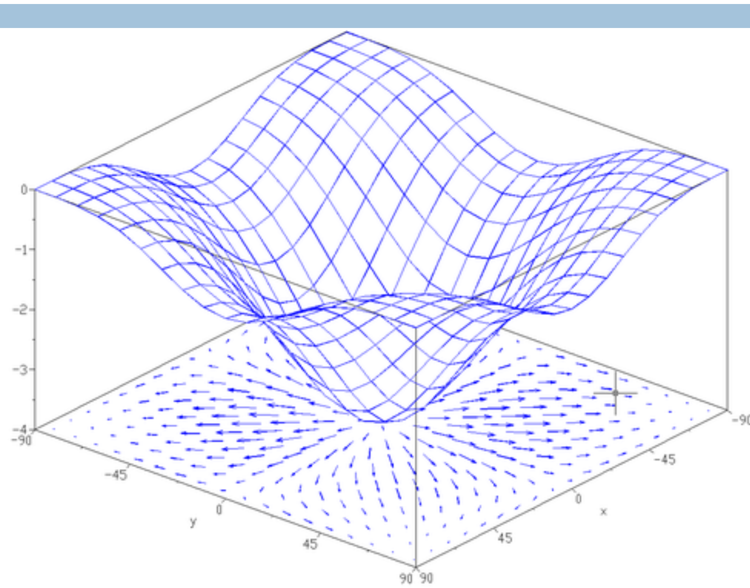


3275411 24748552 32748552

Genetic Algorithms

- Contiguous blocks of string representation must correspond to some meaningful information!
- Developing a string rep., crossover, and mutation function not always straightforward (e.g. TSP – give it a try!)
- Crossover can produce an offspring that is in an entirely different area of the search space than either parent
  - Sometimes offspring is outside of the “feasible” or “evaluable” region

Gradient-based methods



Gradient-based methods

"Their operation is similar to a blind man walking up a hill, whose only knowledge of the hill comes from what passes under his feet. If the hill is predictable in some fashion, he will reach the top, but it is easy to imagine confounding terrain"

- Goffe et al., 1994

Gradient-based methods

- Gradient-based methods
  - Continuous function
  - First derivative easily computable
- Newton-Rhapson applied to optimization
  - Requires computing the first- *and* second-derivative
- Gradient ascent/descent
  - The poor man's version of NR

Gradient Ascent (Descent)

function GRADIENT-ASCENT(F, γ) returns solution

$\nabla F \leftarrow \text{COMPUTE-GRADIENT}(F)$

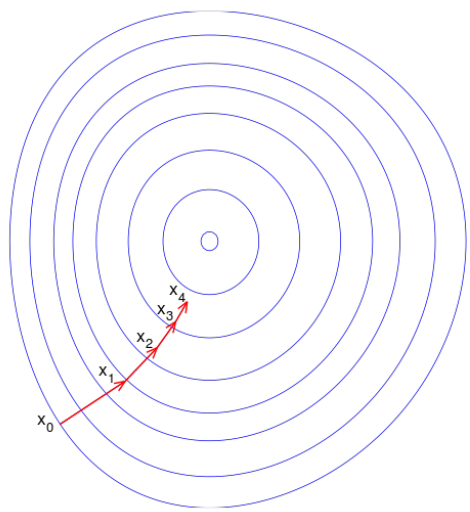
$x \leftarrow$ a randomly selected value

while stopping criteria

$x \leftarrow x + \gamma \nabla F(x)$

return x

Gradient Ascent (Descent)



Local search summary

- Hill-climbing search
  - Stochastic hill-climbing search
  - First-choice
  - Random restart hill-climbing
- Local beam search
  - Stochastic local beam search
- Simulated Annealing
- Genetic algorithms
- Gradient-based methods
  - Newton-Rhapson
  - Gradient ascent (descent)

Read on your own! (Hill-climbing search through Stochastic local beam search)