CSE 473: Artificial Intelligence
Autumn 2016

Local Search

Dan Weld

With slides from
Dan Klein, Stuart Russell, Andrew Moore, Luke Zettlemoyer

Previous Search Methods

Systematic

- Blind Search
  - Depth first search
  - Breadth first search
  - Iterative deepening search
  - Uniform cost search
- Informed Search
  - Best First
  - A*
  - Beam Search
  - Hill Climbing

Heuristic =
Estimate of solution cost

Local (Randomized)

Constraint Satisfaction (Factored)

Beam Search

- Idea
  - Best first but only keep N best items on priority queue
- Evaluation
  - Complete?
  - Time Complexity?
  - Space Complexity?

Hill Climbing

"Gradient ascent"

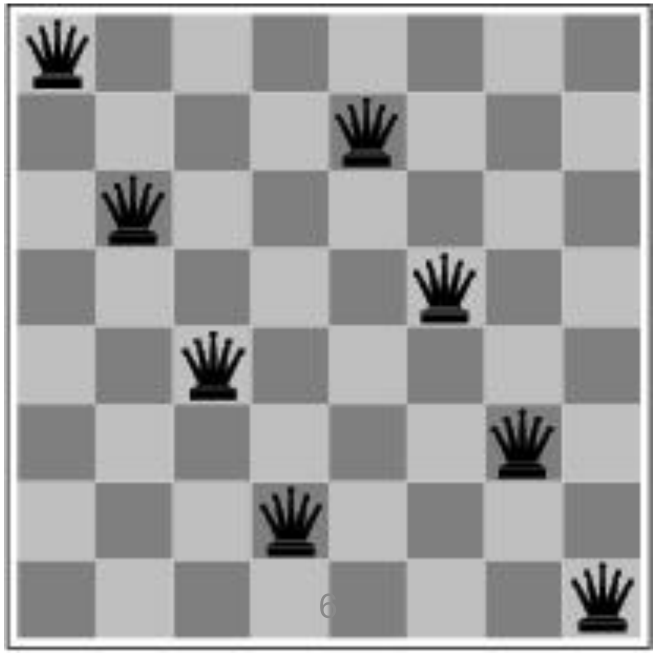
- Idea
  - Always choose best child; no backtracking
  - Beam search with |queue| = 1
- Problems?
  - Coming soon

Goal **State** *vs.* Path

- Previously: Search to find best path to goal
  - Systematic exploration of search space.

- Today: a state is solution to problem
  - For some problems path is irrelevant.
  - E.g., 8-queens

- Different algorithms can be used
  - Systematic Search
  - Local Search
  - Constraint Satisfaction



Local search algorithms

- State space = set of "complete" configurations
- Find configuration satisfying constraints,
  - e.g., all n-queens on board, no attacks
- In such cases, we can use local search algorithms
- Keep a single "current" state, try to improve it.
- Very memory efficient
  - *duh* - only remember current state

Goal Satisfaction

Constraint satisfaction
reach the goal node
guided by heuristic fn

Optimization

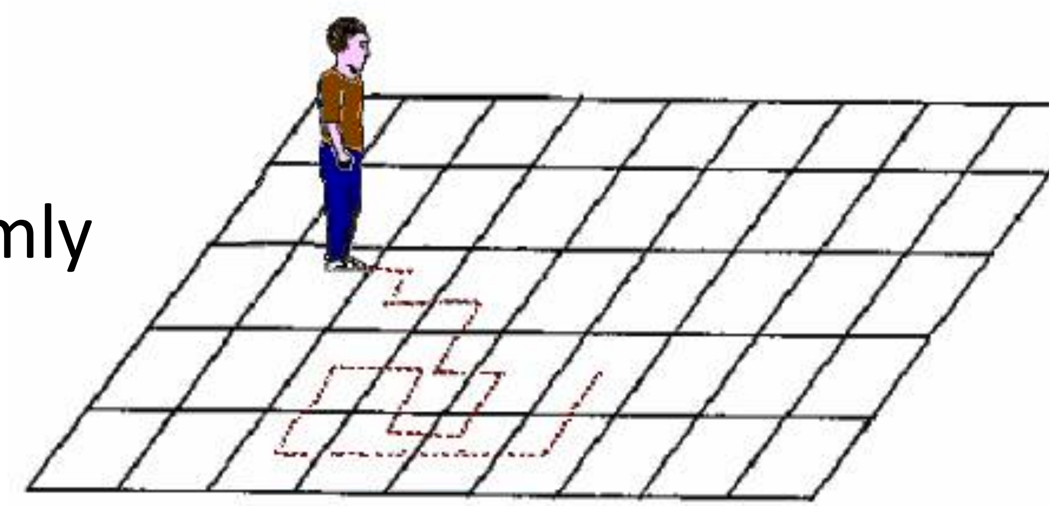
Constraint Optimization
optimize(objective fn)

You can go back and forth between the two problems
Typically in the same complexity class

Local Search and Optimization

- Local search
  - Keep track of single current state
  - Move only to “neighboring” state
    - Defined by operators
  - Ignore previous states, path taken
- Advantages:
  - Use very little memory
  - Can often find reasonable solutions in large or infinite (continuous) state spaces.
- “Pure optimization” problems
  - All states have an objective function
  - Goal is to find state with max (or min) objective value
  - Does not quite fit into path-cost/goal-state formulation
  - Local search can do quite well on these problems.

Trivial Algorithms



- Random Sampling
  - Generate a state randomly
- Random Walk
  - Randomly pick a neighbor of the current state
- Why even mention these?
  - Both algorithms asymptotically complete.
  - http://projecteuclid.org/download/pdf_1/euclid.aop/1176996718 for Random Walk

Hill-climbing search

- "a loop that continuously moves towards increasing value"
  - terminates when a peak is reached
  - Aka greedy local search
- Value can be either
  - Objective function value
  - Heuristic function value (minimized)
- Hill climbing does not look ahead of the immediate neighbors
- Can randomly choose among the set of best successors
  - if multiple have the best value
- "climbing Mount Everest in a thick fog with amnesia"

Example: *n*-Queens

Objective: Put *n* queens on an *n* x *n* board with no two queens on the same row, column, or diagonal

Is this a satisfaction problem or optimization?

Our n-Queens (Local) Search Space

- State
  - All N queens on the board in some configuration
  - But each in a different column
- Successor function
  - Move single queen to another square in same column.

Need Heuristic Function

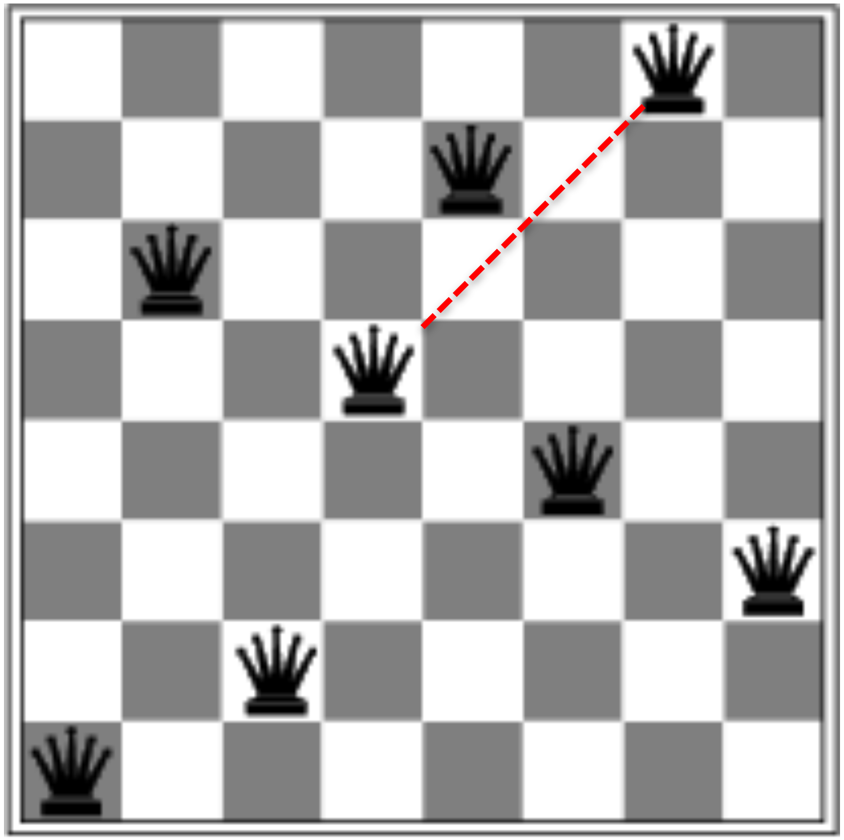
Convert to Optimization Problem

- *h* = number of ***pairs*** of queens attacking each other
- *h = 17* for the above state

Hill-climbing search: 8-queens

Result of hill-climbing in this case…

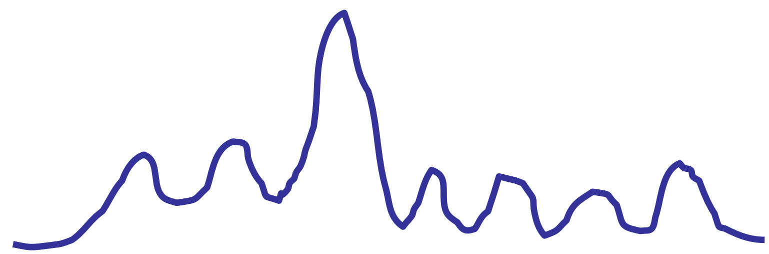
Bummer



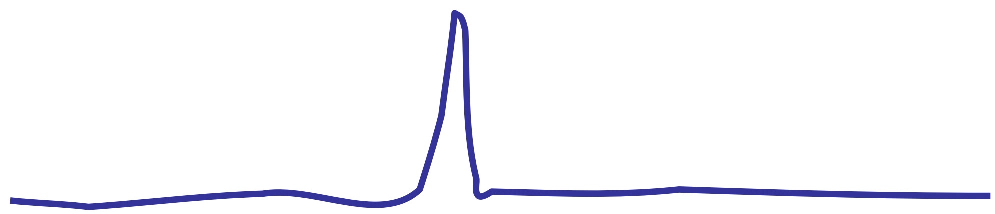
A local minimum with $h = 1$

Hill Climbing Drawbacks

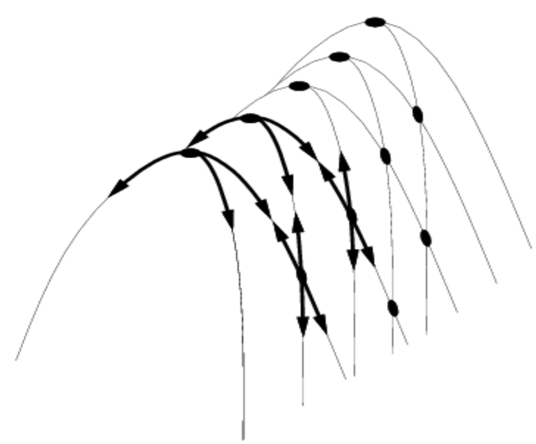
- Local maxima



- Plateaus



- Diagonal ridges



Hill Climbing Properties

- Not Complete
- Worst Case Exponential Time
- Simple, O(1) Space & Often Very Fast!

Hill-climbing on 8-Queens

- Randomly generated 8-queens starting states…
- 14% the time it solves the problem
- 86% of the time it get stuck at a local minimum

- However…
  - Takes only 4 steps on average when it succeeds
  - And 3 on average when it gets stuck
  - (for a state space with 8^8 =~17 million states)

Escaping Shoulders: Sideways Move

- If no downhill (uphill) moves, allow sideways moves in hope that algorithm can escape
  - Must limit the number of possible sideways moves to avoid infinite loops
- For 8-queens
  - Allow sideways moves with limit of 100
  - Raises percentage of problems solved from 14 to 94%

  - However….
    - 21 steps for every successful solution
    - 64 for each failure

Tabu Search

- Prevent returning quickly to the same state
- Keep fixed length queue ("tabu list")
- Add most recent state to queue; drop oldest
- Never move to a tabu state

- Properties:
  - As the size of the tabu list grows, hill-climbing will asymptotically become "non-redundant" (won't look at the same state twice)
  - In practice, a reasonable sized tabu list (say 100 or so) improves the performance of hill climbing in many problems

Escaping Local Optima - Enforced Hill Climbing

- Perform breadth first search from a local optima
  - to find the next state with better h function
- Typically,
  - prolonged periods of exhaustive search
  - bridged by relatively quick periods of hill-climbing
- Middle ground b/w local and systematic search

Hill Climbing: Stochastic Variations

→When the state-space landscape has local minima, any search that moves only in the greedy direction cannot be complete

→Random walk, on the other hand, *is* asymptotically complete

Idea: Combine random walk & greedy hill-climbing

At each step do one of the following:

- Greedy: With prob p move to the neighbor with largest value
- Random: With prob 1-p move to a random neighbor

Hill-climbing with random restarts

- If at first you don't succeed, try, try again!
- Different variations
  - For each restart: run until termination vs. run for a fixed time
  - Run a fixed number of restarts or run indefinitely

Use this algorithm!

- Analysis
  - Say each search has probability p of success
    - E.g., for 8-queens, p = 0.14 with no sideways moves
  - Expected number of restarts?

Restarts	0	2	4	8	16	32	64
Success?	14%	36%	53%	74%	92%	99%	99.994%

  - Expected number of steps taken?

Hill-Climbing with Both Random Walk & Random Sampling

At each step do one of the three

- Greedy: move to the neighbor with largest value
- Random Walk: move to a random neighbor
- Random Restart: Start over from a new, random state

Simulated Annealing
written to find minimum value solutions

function SIMULATED-ANNEALING(*problem, schedule*) **return** a solution state

input: *problem*, a problem

schedule, a mapping from time to temperature

local variables: *current***,** a node.

*next***,** a node.

*T***,** a "temperature" controlling the prob. 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 $e^{-\Delta E/T}$

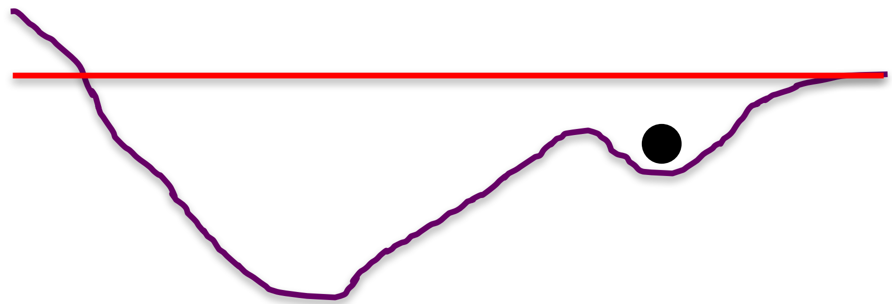
Physical Interpretation of Simulated Annealing

Minimization (not max)

- A Physical Analogy:
  - Imagine letting a ball roll downhill on the function surface
  - Now shake the surface, while the ball rolls,
  - Gradually reducing the amount of shaking

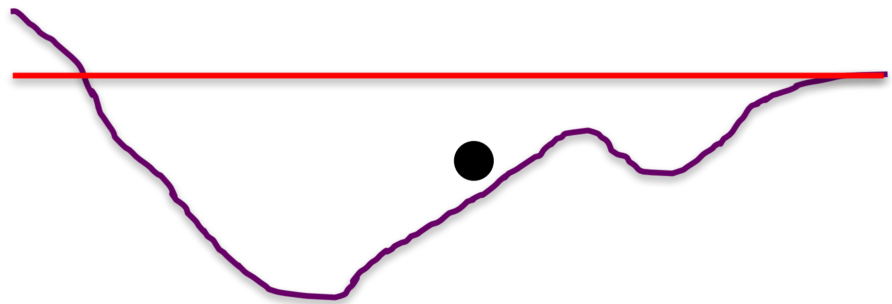
Physical Interpretation of Simulated Annealing

- A Physical Analogy:
  - Imagine letting a ball roll downhill on the function surface
  - Now shake the surface, while the ball rolls,
  - Gradually reducing the amount of shaking



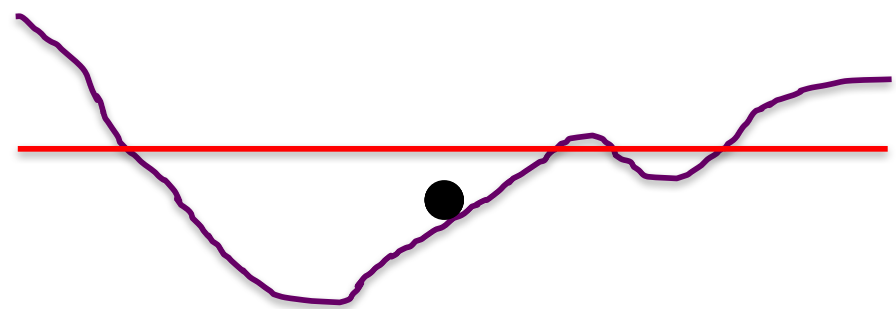
Physical Interpretation of Simulated Annealing

- A Physical Analogy:
  - Imagine letting a ball roll downhill on the function surface
  - Now shake the surface, while the ball rolls,
  - Gradually reducing the amount of shaking



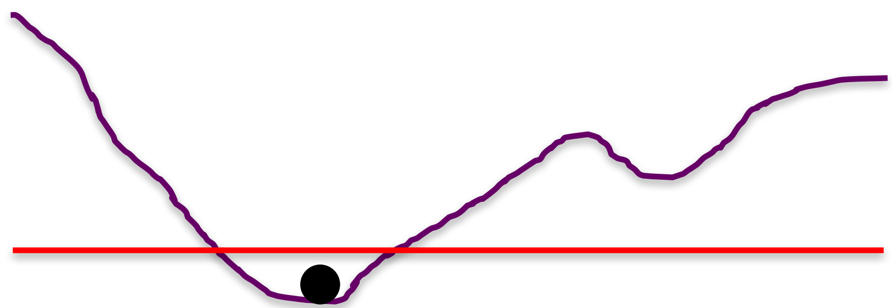
Physical Interpretation of Simulated Annealing

- A Physical Analogy:
  - Imagine letting a ball roll downhill on the function surface
  - Now shake the surface, while the ball rolls,
  - Gradually reducing the amount of shaking



Physical Interpretation of Simulated Annealing

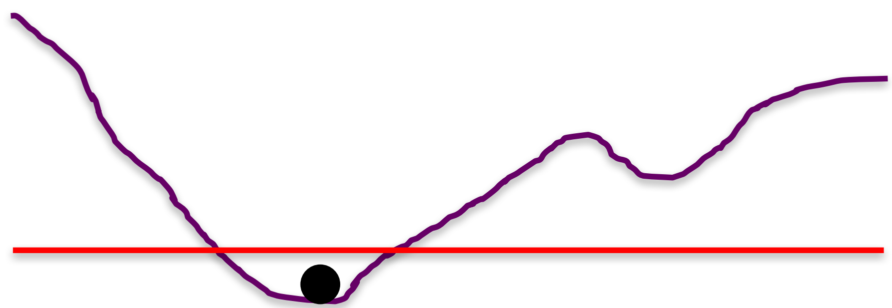
- A Physical Analogy:
  - Imagine letting a ball roll downhill on the function surface
  - Now shake the surface, while the ball rolls,
  - Gradually reducing the amount of shaking



- Annealing = physical process of cooling a liquid → frozen
  - simulated annealing:
    - free variables are like particles
    - seek "low energy" (high quality) configuration
    - slowly reducing temp. T with particles moving around randomly

Temperature T

- high T: probability of "locally bad" move is higher
- low T: probability of "locally bad" move is lower
- typically, T is decreased as the algorithm runs longer
- i.e., there is a "temperature schedule"



Simulated Annealing in Practice

- method proposed in 1983 by IBM researchers for solving VLSI layout problems (Kirkpatrick et al, *Science*, 220:671-680, 1983).
  - theoretically will always find the global optimum
- Other applications: Traveling salesman, Graph partitioning, Graph coloring, Scheduling, Facility Layout, Image Processing, …
- useful for some problems, but can be very slow
  - slowness comes about because T must be decreased very gradually to retain optimality

Local beam search

- Idea: Keeping only one node in memory is an extreme reaction to memory problems.

- Keep track of *k* states instead of one
  - Initially: *k* randomly selected states
  - Next: determine all successors of *k* states
  - If any of successors is goal $\rightarrow$ finished
  - Else select *k* best from successors and repeat

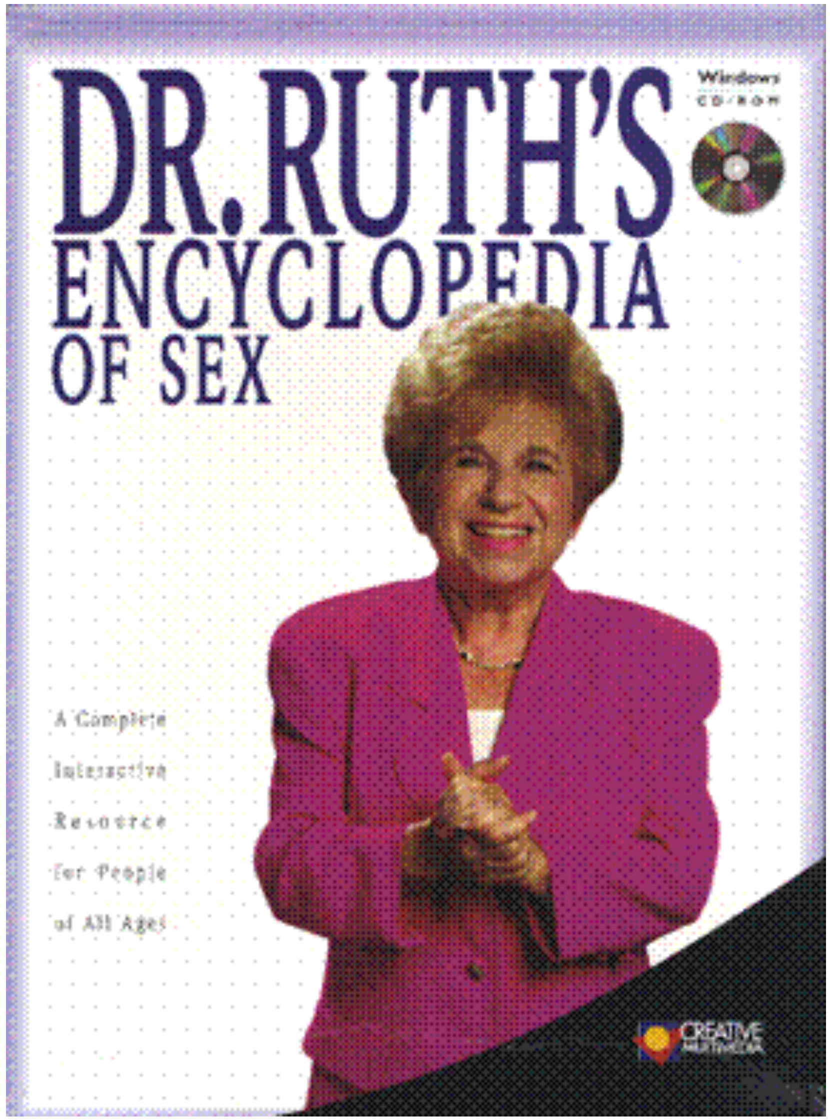
Local Beam Search (contd)

- Not the same as *k random-start searches run in parallel!*
- Searches that find good states recruit other searches to join them

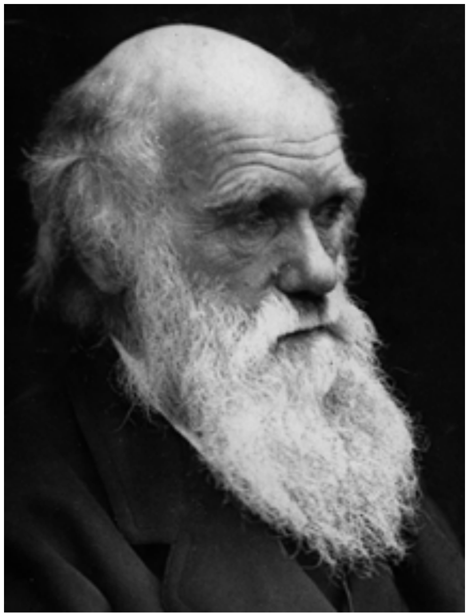
- Problem: quite often, all *k states end up on same local hill*
- Idea: Stochastic beam search
  - Choose *k successors randomly, biased towards good ones*

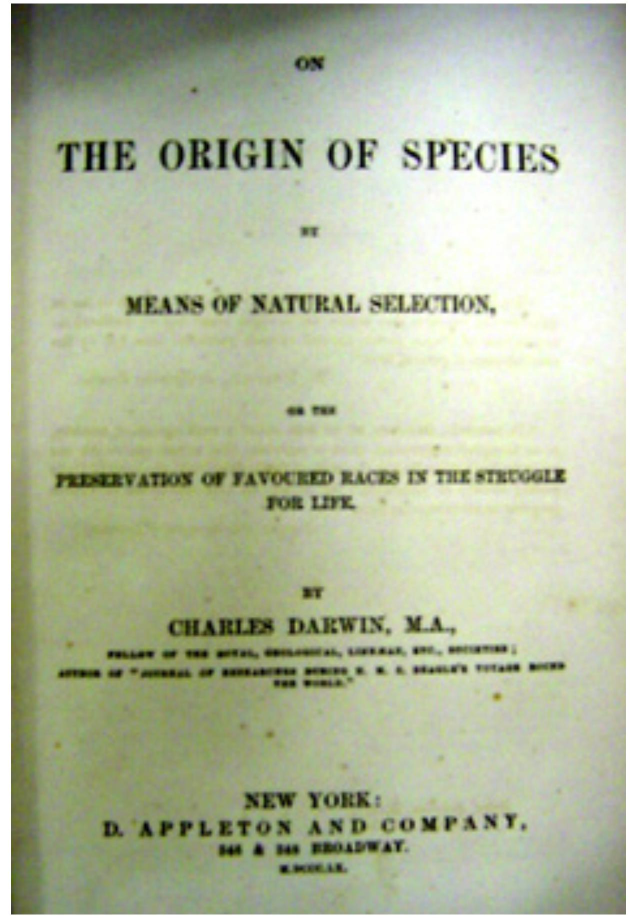
- Observe the close analogy to natural selection!

How to Make Search More … ***Exciting?***



…And Be Scholarly!





ON

THE ORIGIN OF SPECIES

BY

MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE
FOR LIFE.

BY

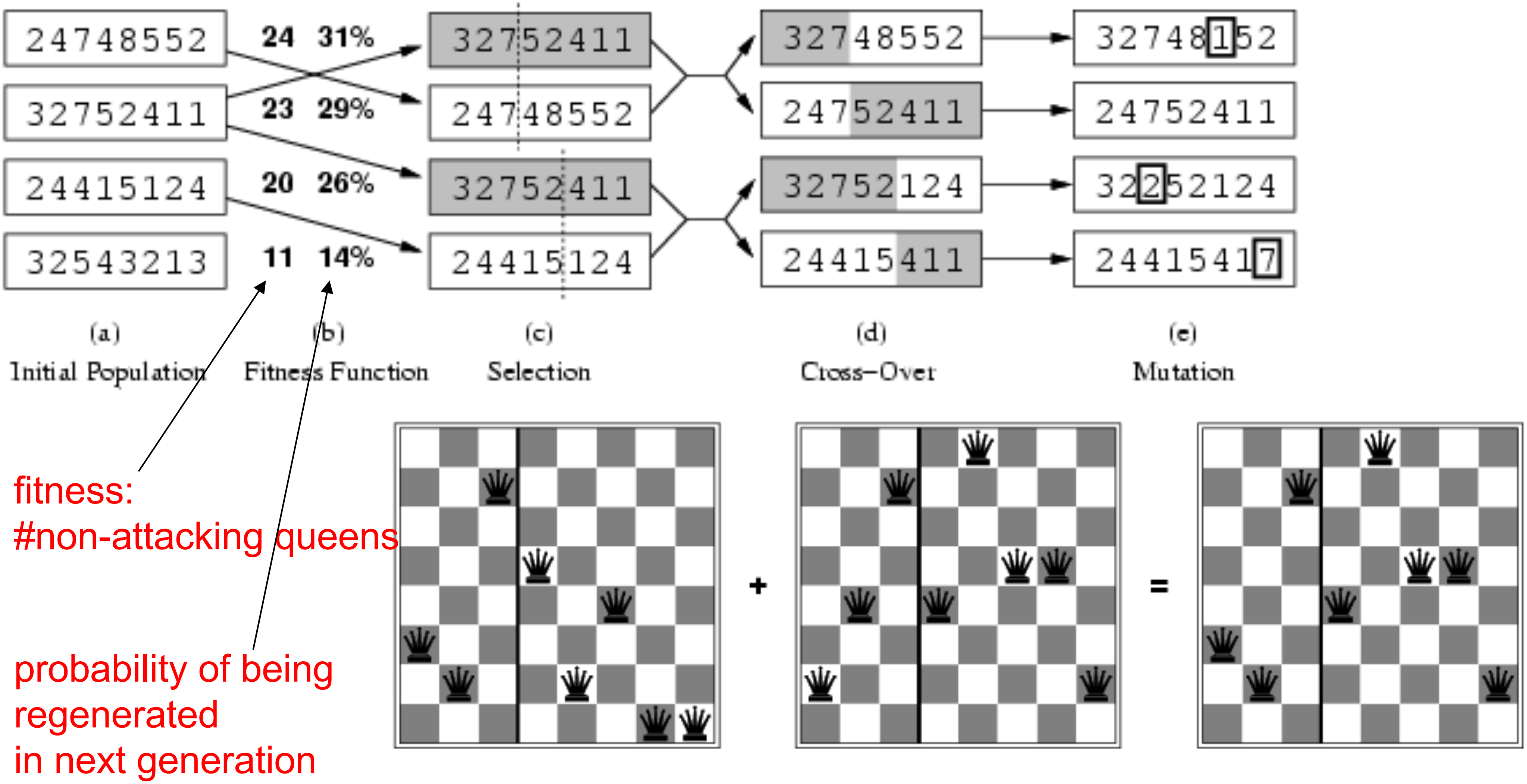
CHARLES DARWIN, M.A.,

[illegible]

NEW YORK:
D. APPLETON AND COMPANY,
[illegible] BROADWAY.
[illegible]

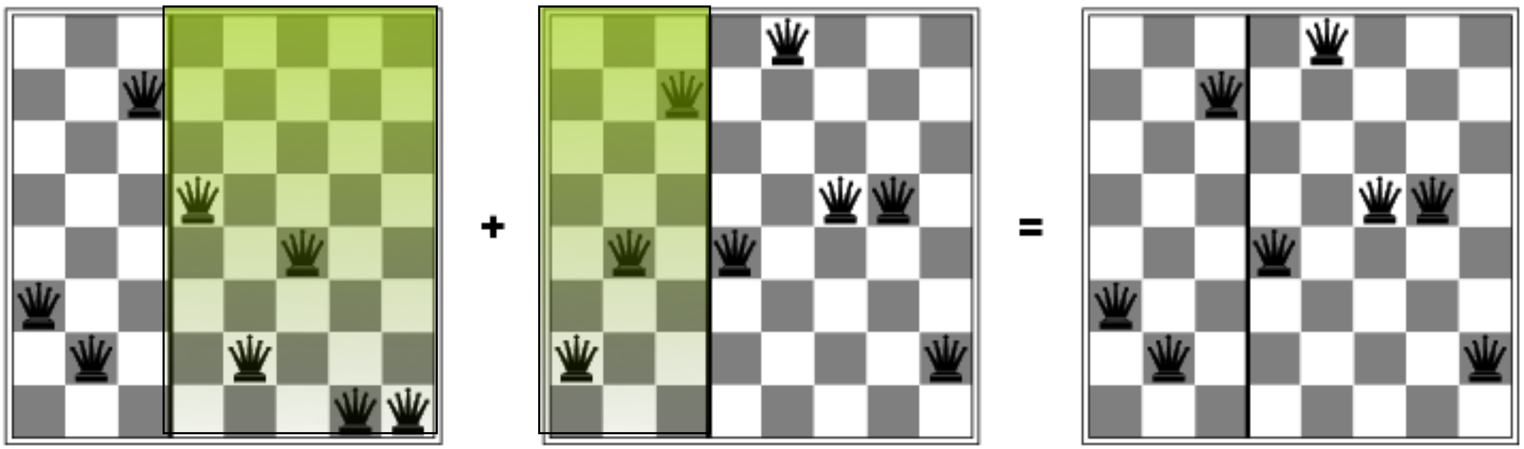
Genetic algorithms

- Local beam search, but…
  - A successor state is generated by ***combining two parent states***
- Start with *k* randomly generated states (population)
- A state is represented as a ***string*** over a finite alphabet (often a string of 0s and 1s)
- Evaluation function (fitness function). Higher = better
- Produce the next generation of states by selection, crossover, and mutation



- Fitness function: number of non-attacking pairs of queens (min = 0, max = 8 × 7/2 = 28)
- 24/(24+23+20+11) = 31%
- 23/(24+23+20+11) = 29% etc

Genetic algorithms



Has the effect of "jumping" to a completely different new part of the search space (quite non-local)

Comments on Genetic Algorithms

- Genetic algorithm is a variant of "stochastic beam search"
- Positive points
  - Random exploration can find solutions that local search can't
    - (via crossover primarily)
  - Appealing connection to human evolution
    - "neural" networks, and "genetic" algorithms are **metaphors**!
- Negative points
  - Large number of "tunable" parameters
    - Difficult to replicate performance from one problem to another
  - Lack of good empirical studies comparing to simpler methods
  - Useful on some (small?) set of problems but no convincing evidence that GAs are better than hill-climbing w/random restarts in general