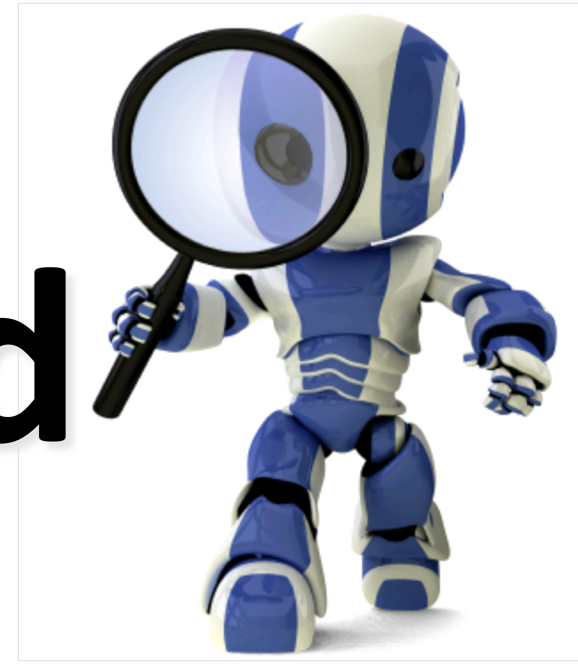
Informed Search

Chapter 4 (b)



Some material adopted from notes by Charles R. Dyer, University of Wisconsin-Madison

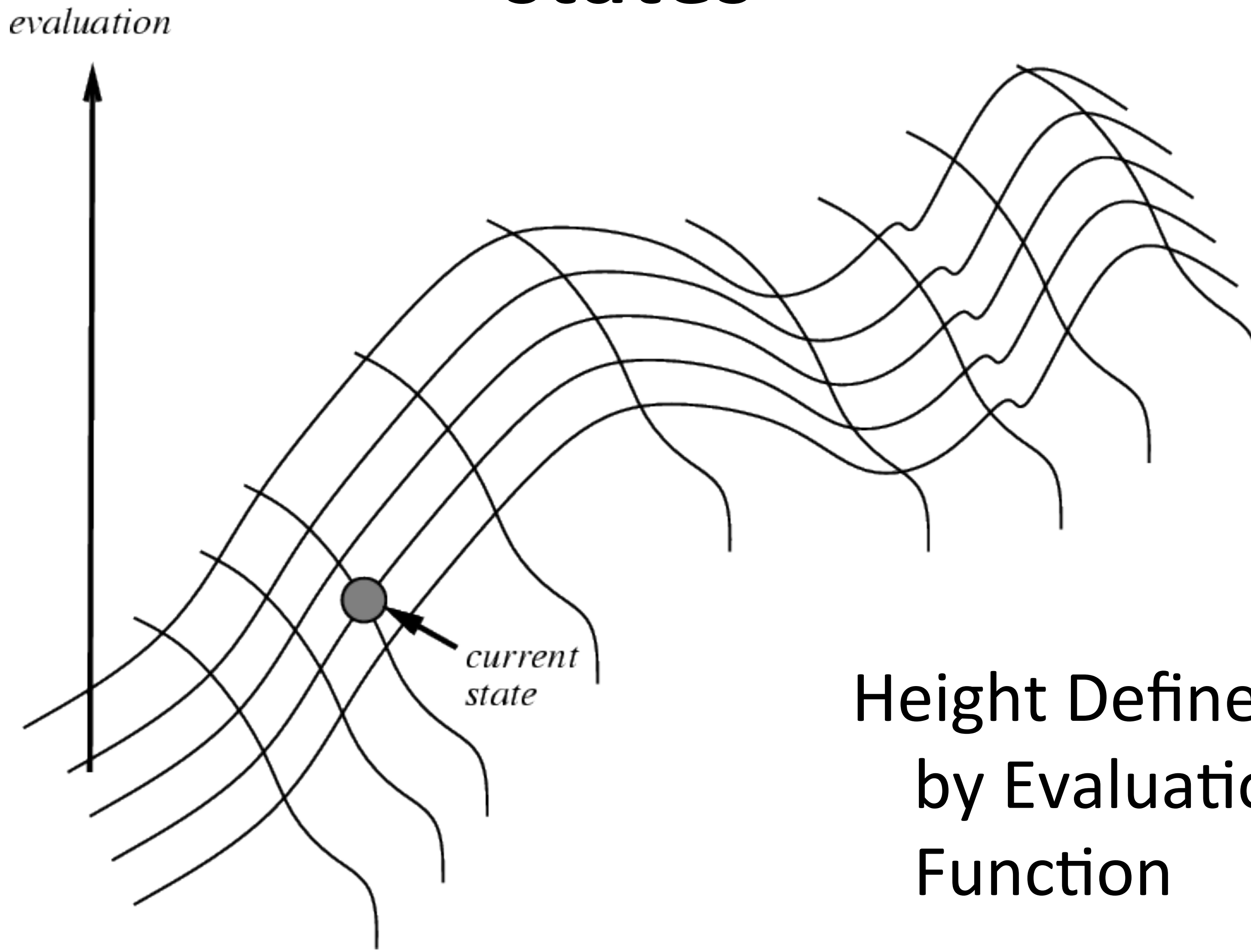
Today's class: local search

- Iterative improvement methods
  - Hill climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
- Online search

Hill Climbing

- Extended current path with successor that's closer to the solution than end of current path
- If goal is to get to the top of a hill, then always take a step the leads you up
- Simple hill climbing – take any upward step
- Steepest ascent hill climbing – consider all possible steps and take one that goes up the most
- No memory

Hill climbing on a surface of states



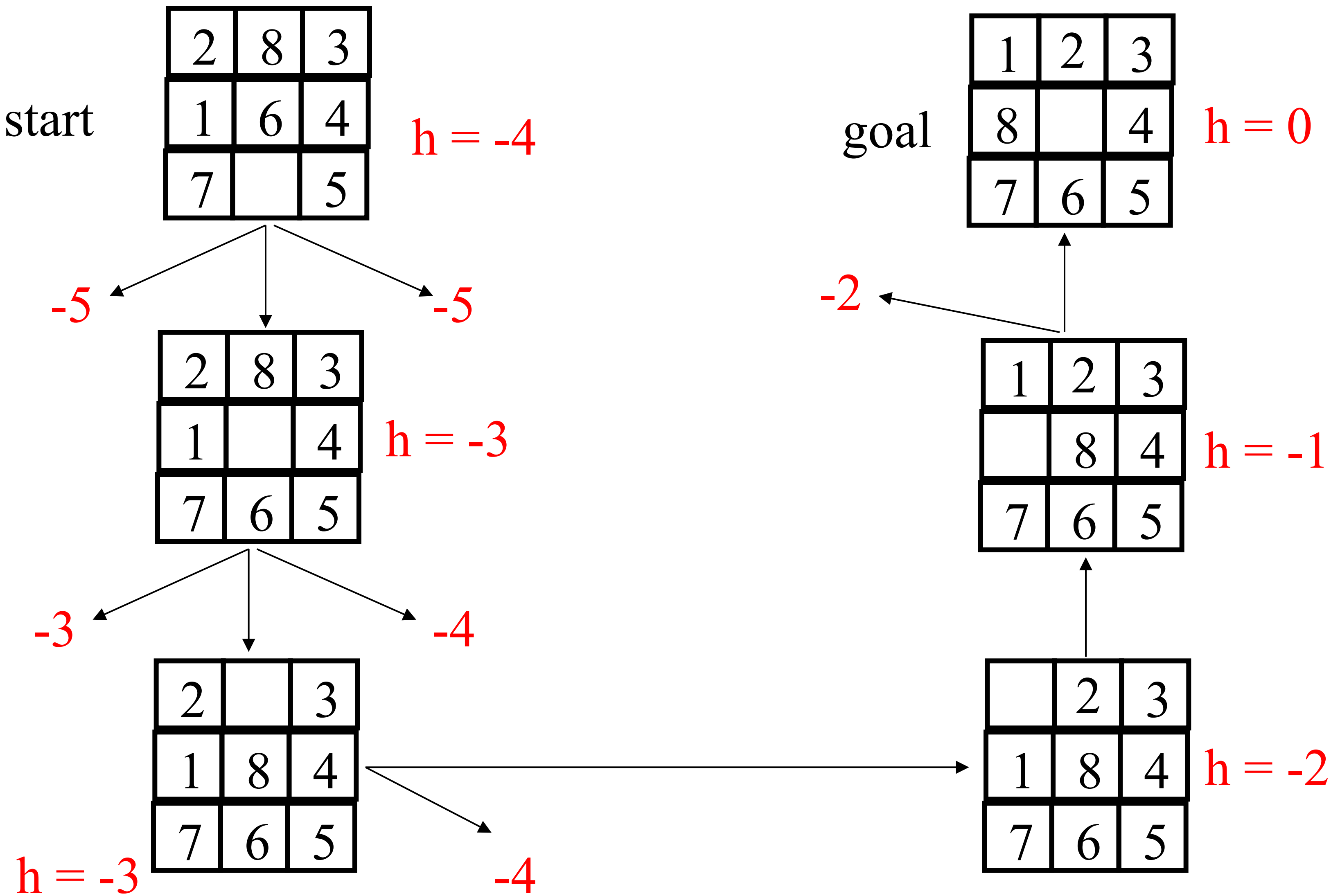
Height Defined by Evaluation Function

Hill-climbing search

- If there's successor s for current state n such that
  - $h(s) < h(n)$ and $h(s) <= h(t)$ for all successors t

  then move from n to s. Otherwise, halt at n
- Look 1 step ahead to decide if a successor is better than current state; if so, move to best successor
- Like Greedy search, but doesn't allow backtracking or jumping to alternative path since it has no memory
- Like beam search with a beam width of 1 (i.e., the maximum size of the nodes list is 1)
- Not complete since the search will terminate at "local minima", "plateaus," and "ridges"

Hill climbing example



f(n) = -(number of tiles out of place)

Exploring the Landscape

- **Local Maxima**: peaks that aren't highest point in space
- **Plateaus:** space has a broad flat region that gives search algorithm no direction (random walk)
- **Ridges:** flat like plateau, but with drop-offs to sides; steps to North, East, South and West may go down, but step to NW may go up

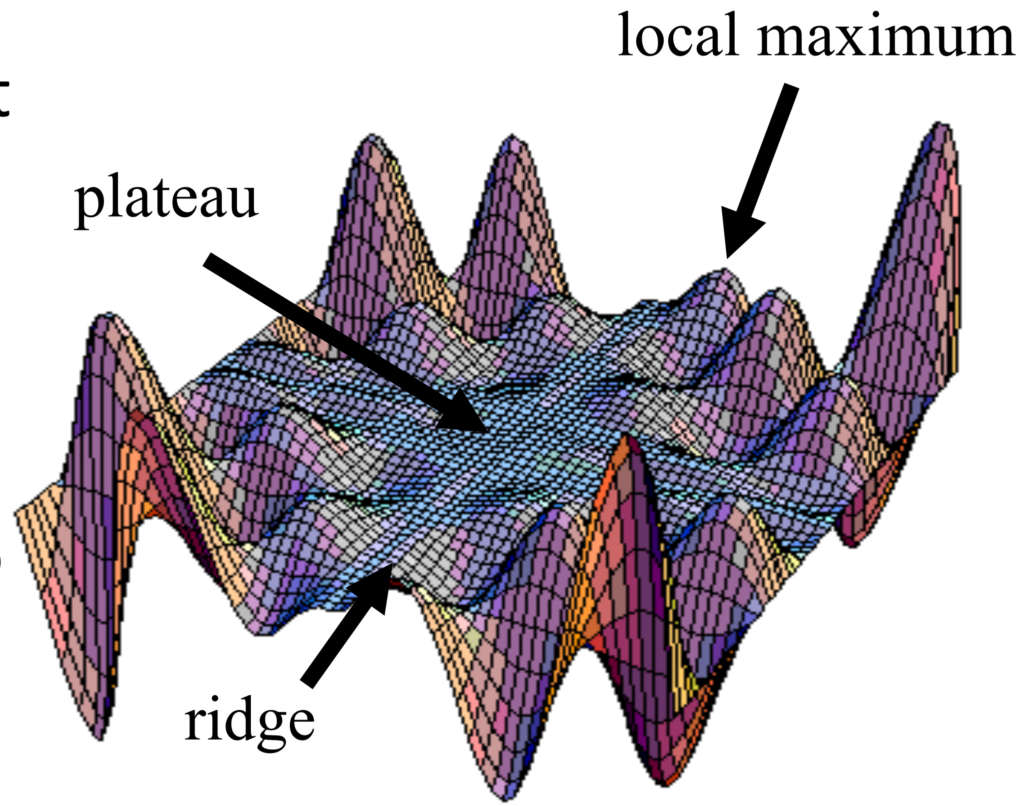
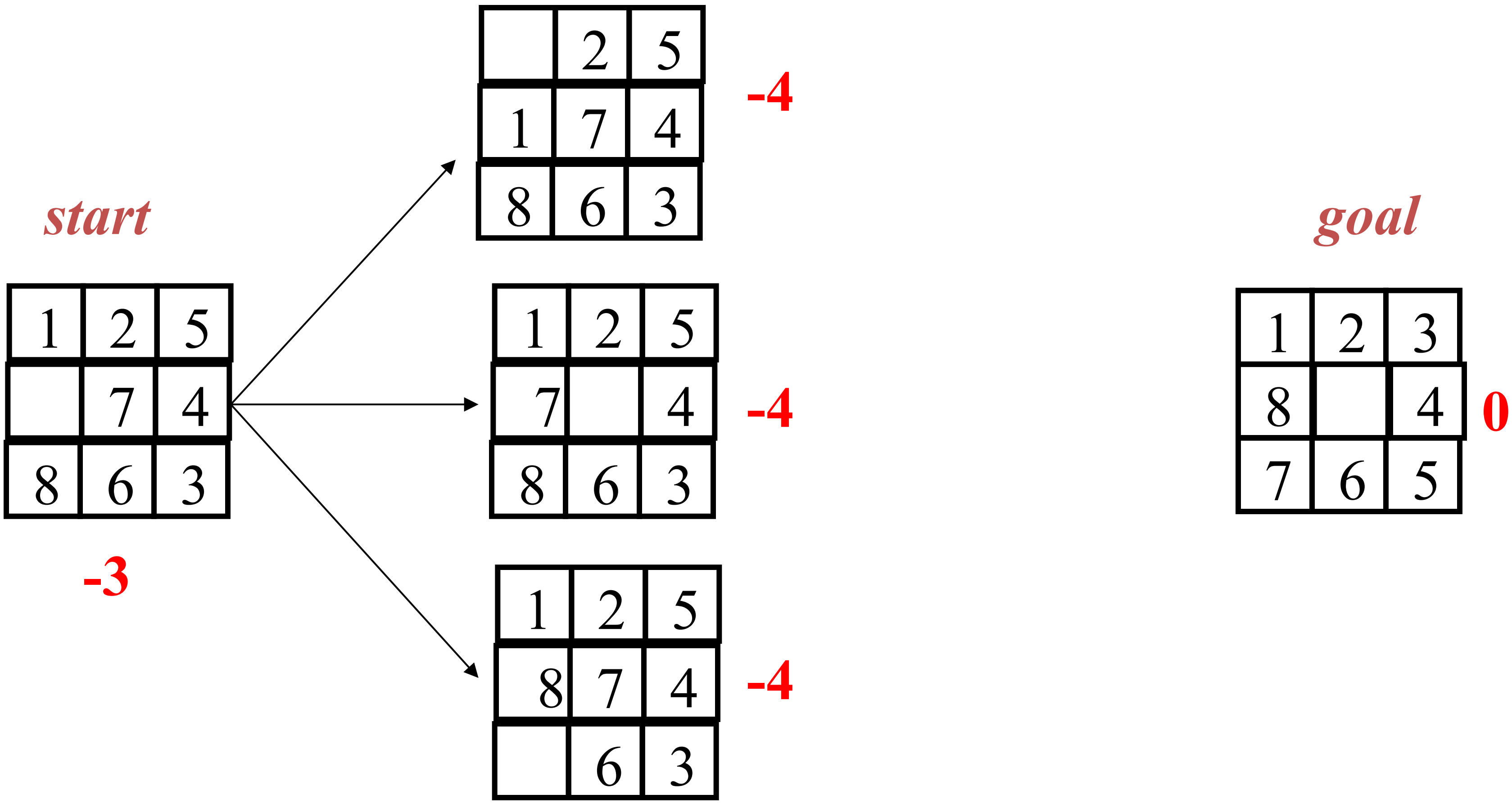


Image from: http://classes.yale.edu/fractals/CA/GA/Fitness/Fitness.html

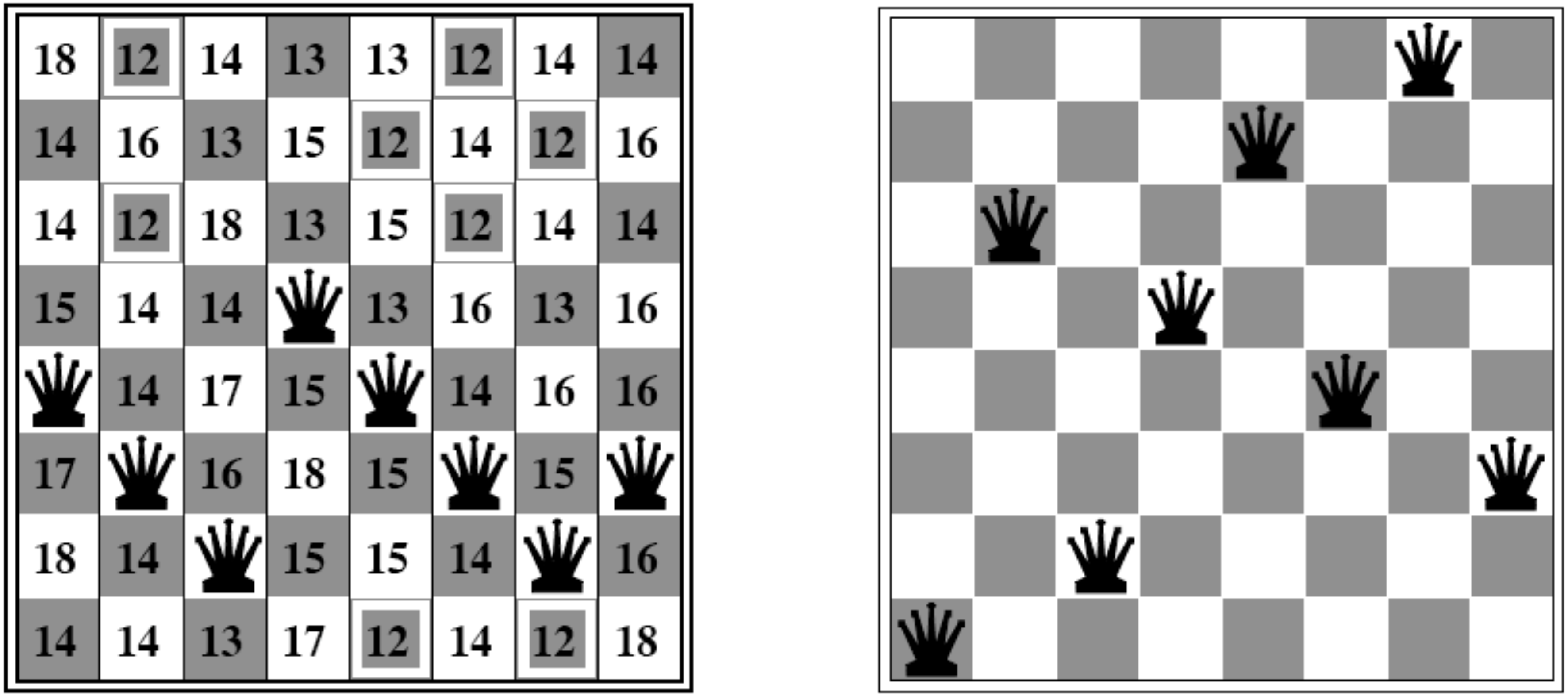
Drawbacks of hill climbing

- Problems: local maxima, plateaus, ridges
- Remedies:
  - **Random restart:** keep restarting the search from random locations until a goal is found
  - **Problem reformulation:** reformulate the search space to eliminate these problematic features
- Some problem spaces are great for hill climbing and others are terrible

Example of a local optimum



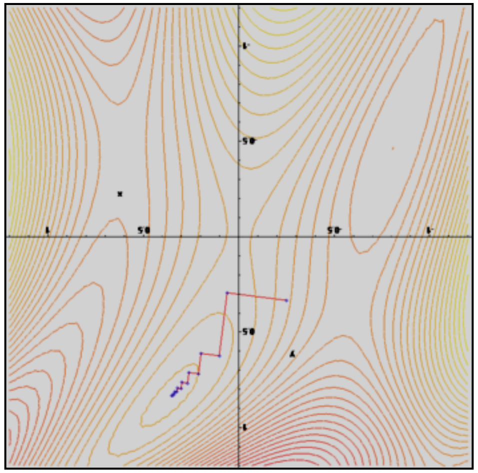
Hill Climbing and 8 Queens

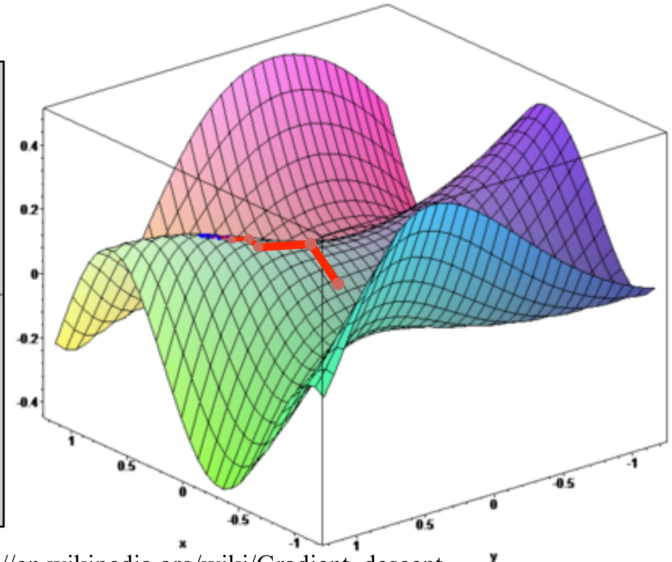


(a) (b)

Figure 4.3 (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

Gradient ascent / descent





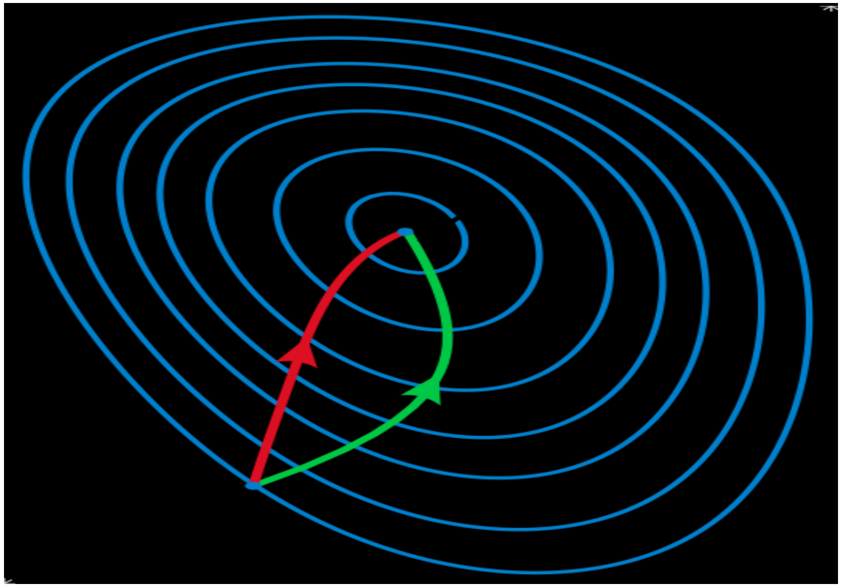
Images from http://en.wikipedia.org/wiki/Gradient_descent

- Gradient descent procedure for finding the $arg_x\ min\ f(x)$
  - choose initial x_0 randomly
  - repeat
    - $x_{i+1} \leftarrow x_i - \eta f'(x_i)$
  - until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
- Step size η (eta) is small (perhaps 0.1 or 0.05)

Gradient methods vs. Newton's method

- A reminder of Newton's method from Calculus:

  $x_{i+1} \leftarrow x_i - \eta f'(x_i) / f''(x_i)$
- Newton's method uses 2nd order information (second derivative, or, curvature) to take a more direct route to the minimum.
- The second-order information is more expensive to compute, but converges quicker.



Contour lines of a function
Gradient descent (green)
Newton's method (red)
Image from http://en.wikipedia.org/wiki/Newton's_method_in_optimization

Annealing

- In metallurgy, annealing is a technique involving heating and controlled cooling of a material to increase size of its crystals and reduce their defects
- Heat causes atoms to become unstuck from initial positions (local minima of internal energy) and wander randomly through states of higher energy
- Slow cooling gives them more chances of finding configurations with lower internal energy than initial one

Simulated annealing (SA)

- SA exploits the analogy between how metal cools and freezes into a minimum-energy crystalline structure & search for a minimum/maximum in a general system
- SA can avoid becoming trapped at local minima
- SA uses a random search that accepts changes increasing objective function f and some that **decrease** it
- SA uses a control parameter T, which by analogy with the original application is known as the system "**temperature**"
- T starts out high and gradually decreases toward 0

SA intuitions

- Combines hill climbing (efficiency) with random walk (completeness)
- Analogy: getting a ping-pong ball into the deepest depression in a bumpy surface
  - shake the surface to get the ball out of the local minima
  - not too hard to dislodge it from the global minimum
- Simulated annealing:
  - start by shaking hard (high temperature) and gradually reduce shaking intensity (lower the temperature)
  - escape the local minima by allowing some "bad" moves
  - but gradually reduce their size and frequency

Simulated annealing

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

$$e^{-(f(B)-f(A)/T)}$$

- 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 admissible

Simulated annealing algorithm

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

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

Figure 4.5 The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed. Downhill moves are accepted readily early in the annealing schedule and then less often as time goes on. The *schedule* input determines the value of T as a function of time.

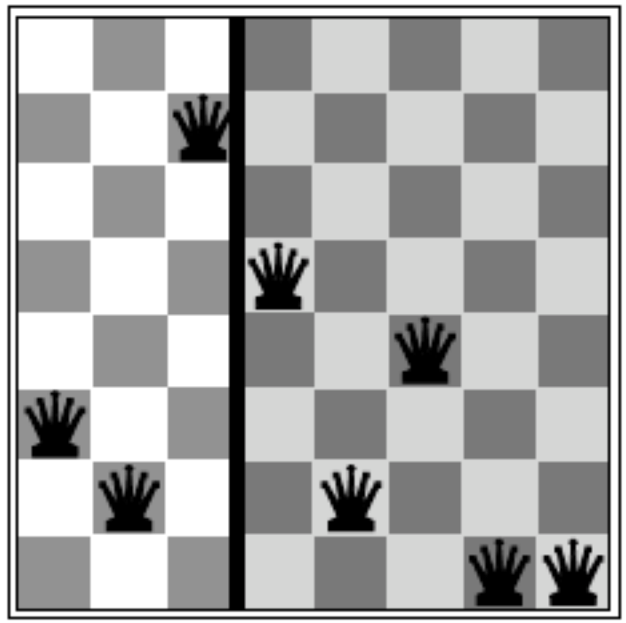
Local beam search

- Basic idea
  - Begin with k random states
  - Generate all successors of these states
  - Keep the k best states generated by them
- Provides a simple, efficient way to share some knowledge across a set of searches
- *Stochastic beam search* is a variation:
  - Probability of keeping a state is *a function* of its heuristic value

Genetic algorithms (GA)

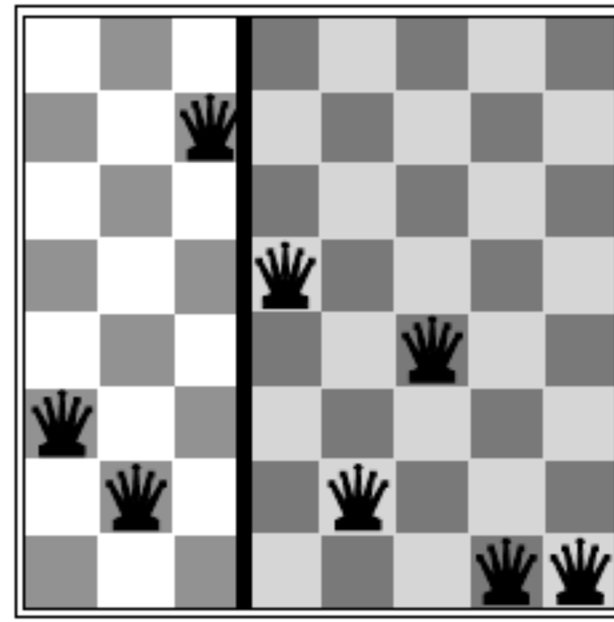
- A search technique inspired by *evolution*
- Similar to stochastic beam search
- Start with k random states (the *initial population)*
- New states are generated by "mutating" a single state or "reproducing" (combining) two parent states (selected according to their *fitness)*
- Encoding used for the "genome" of an individual strongly affects the behavior of the search
- Genetic algorithms / genetic programming are a large and active area of research

Ma and Pa solutions



8 Queens problem

- Represent state by a string of 8 digits in {1..8}
- S = '32752411'
- Fitness function = # of non-attacking pairs
- $F(S_{solution}) = 8*7/2 = 28$
- $F(S_1) = 24$



Genetic algorithms

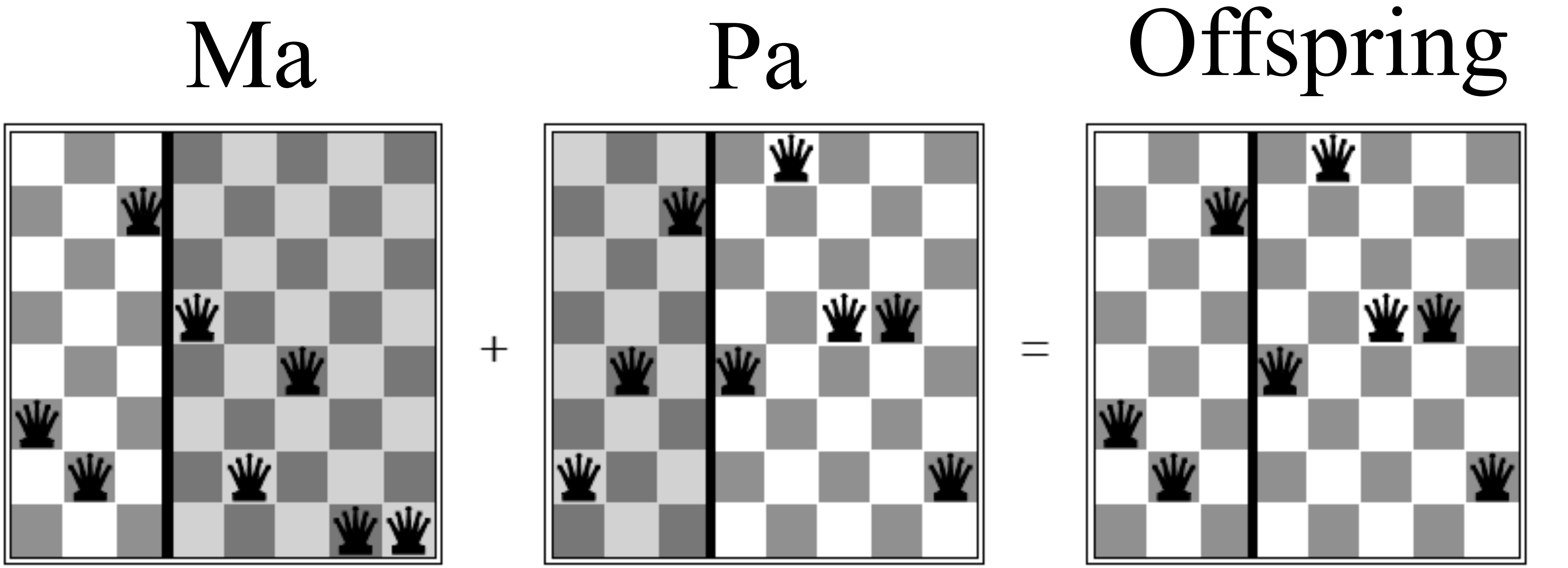


Figure 4.7 The 8-queens states corresponding to the first two parents in Figure 4.6(c) and the first offspring in Figure 4.6(d). The shaded columns are lost in the crossover step and the unshaded columns are retained.

Genetic algorithms

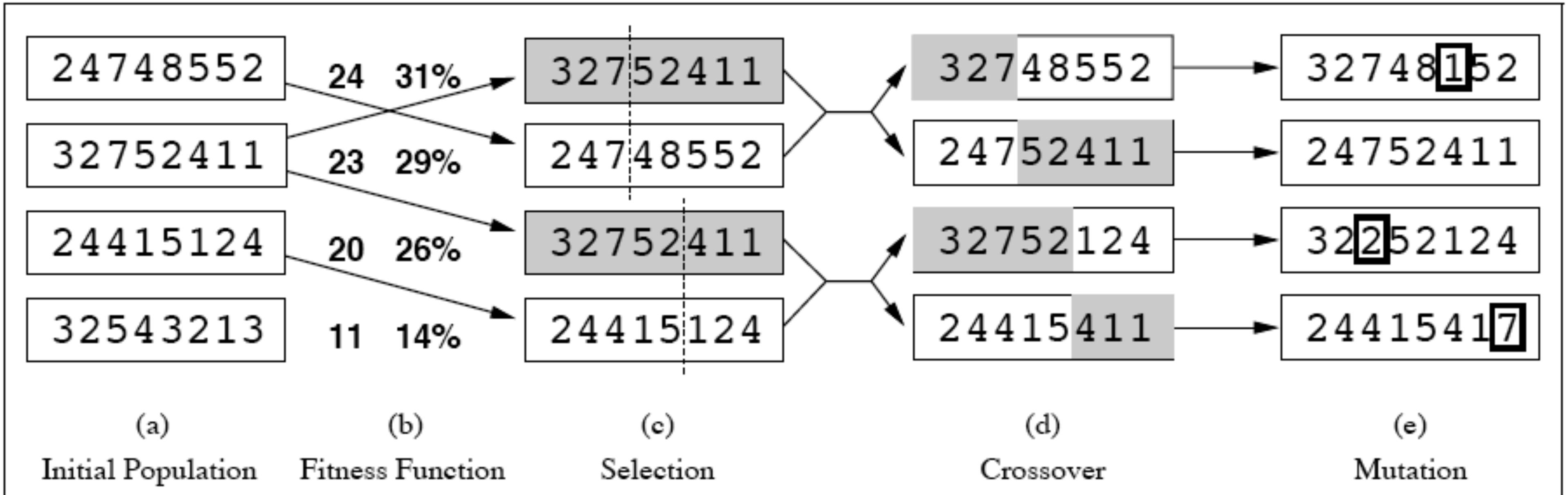


Figure 4.6 The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

- 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

GA pseudo-code

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
     new_population ← empty set
     for i = 1 to SIZE(population) do
        x ← RANDOM-SELECTION(population, FITNESS-FN)
        y ← RANDOM-SELECTION(population, FITNESS-FN)
        child ← REPRODUCE(x, y)
        if (small random probability) then child ← MUTATE(child)
        add child to new_population
     population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

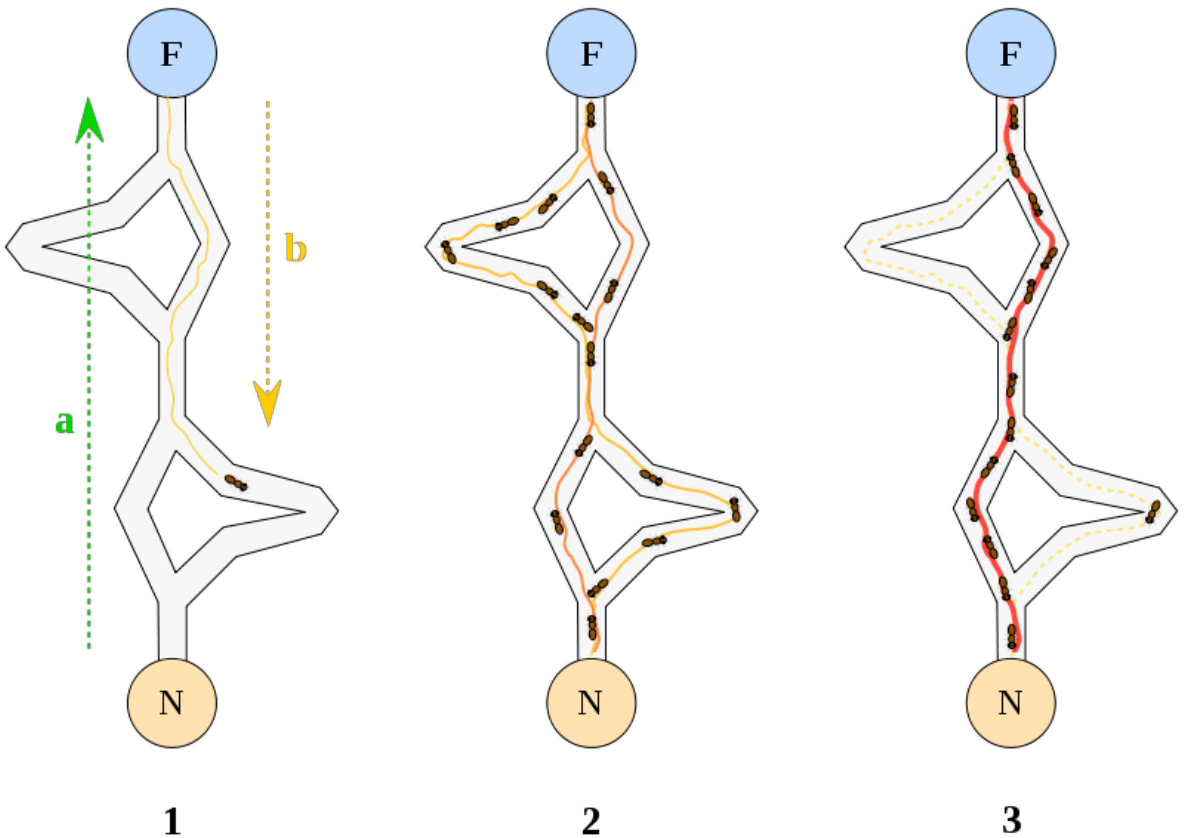
```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Ant Colony Optimization

A probabilistic search technique for problems reducible to finding good paths through graphs

Inspiration

- Ants leave nest
- Discover food
- Return to nest, preferring shorter paths
- Leave pheromone trail
- Shortest path is reinforced



An example of agents communicating through their environment

Tabu search

- 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

CLASS EXERCISE

- What would a local search approach to solving a Sudoku problem look like?

	3		
			1
3			
		2	

Online search

- Interleave computation & action
  - search some, act some
- Exploration: Can't infer outcomes of actions; must actually perform them to learn what will happen
- 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 these in chapters on Logic and Learning!

Other topics

- Search in continuous spaces
  - Different math
- Search with uncertain actions
  - Must model the probabilities of an actions results
- Search with partial observations
  - Acquiring knowledge as a result of search

Summary: Informed 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.