Informed Search

Chapter 4 (b)

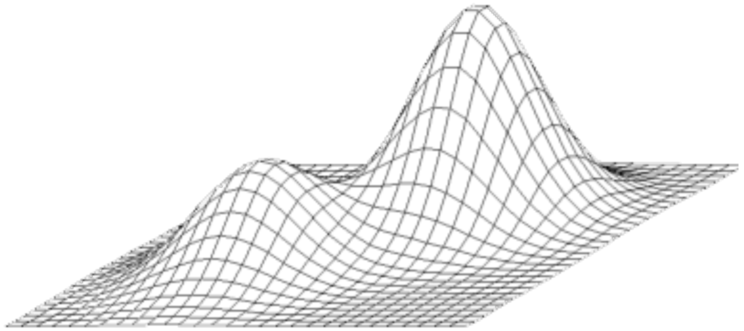
Some material adopted from notes by Charles R. Dyer, University of Wisconsin-Madison

Today’s class: local search

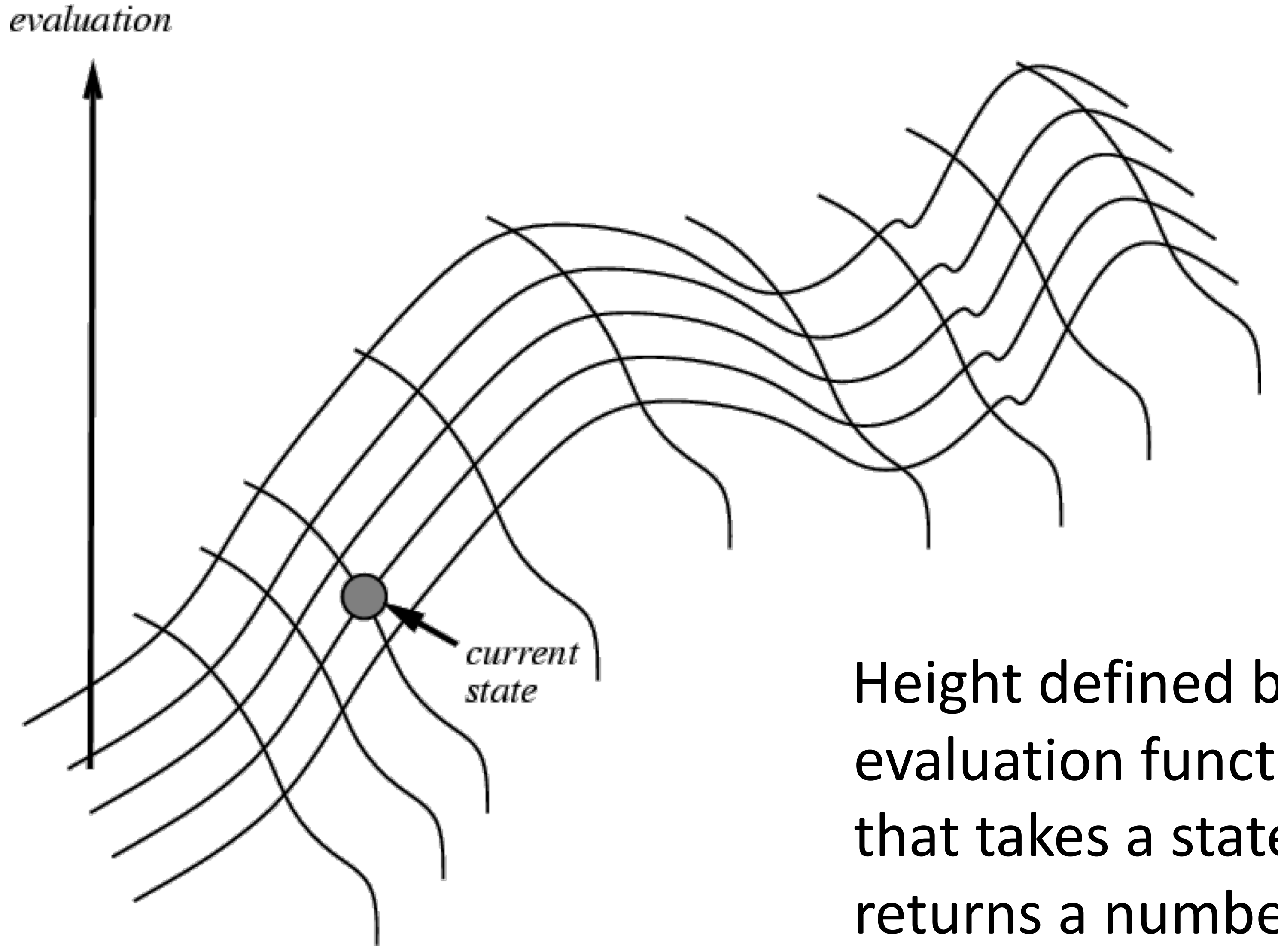
- Iterative improvement methods (aka local search) move from potential solution to potential solution until a goal is reached
- Examples
  - Hill climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
- Online search

Hill Climbing

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



Hill climbing on a surface of states



Height defined by an evaluation function that takes a state & returns a number

Hill climbing for search

- For informed search and many other problems (e.g., neural network training) we want to find a **global minimum**
  - Search evaluation function: measure of how far the current state is from a goal
- It’s an easy change to make in the algorithm, or we can just negate the evaluation function
- We still call it hill *climbing* though

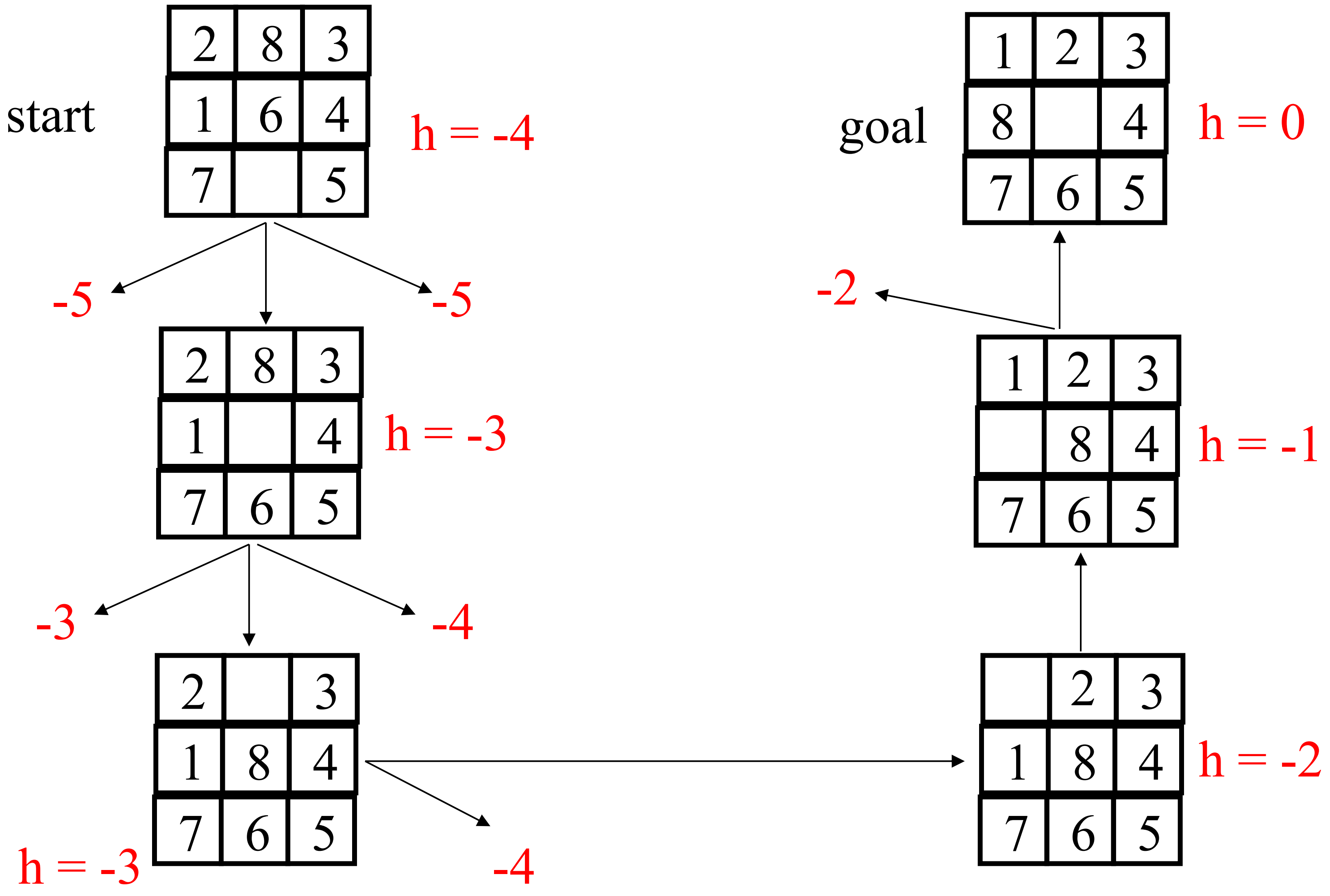
Hill-climbing search

- 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**

  i.e.: Look one 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., maximum size of the nodes list is 1)
- Not complete since search may terminate at a local minima, plateau or ridge

Hill climbing example



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

Exploring the Landscape

- **Local Maxima**: peaks not highest point in space
- **Plateaus:** broad flat region giving search no guidance (use random walk)
- **Ridges:** flat like plateaus, but with drop-offs to sides; steps to North, East, South and West may go down, but step to NW may go up

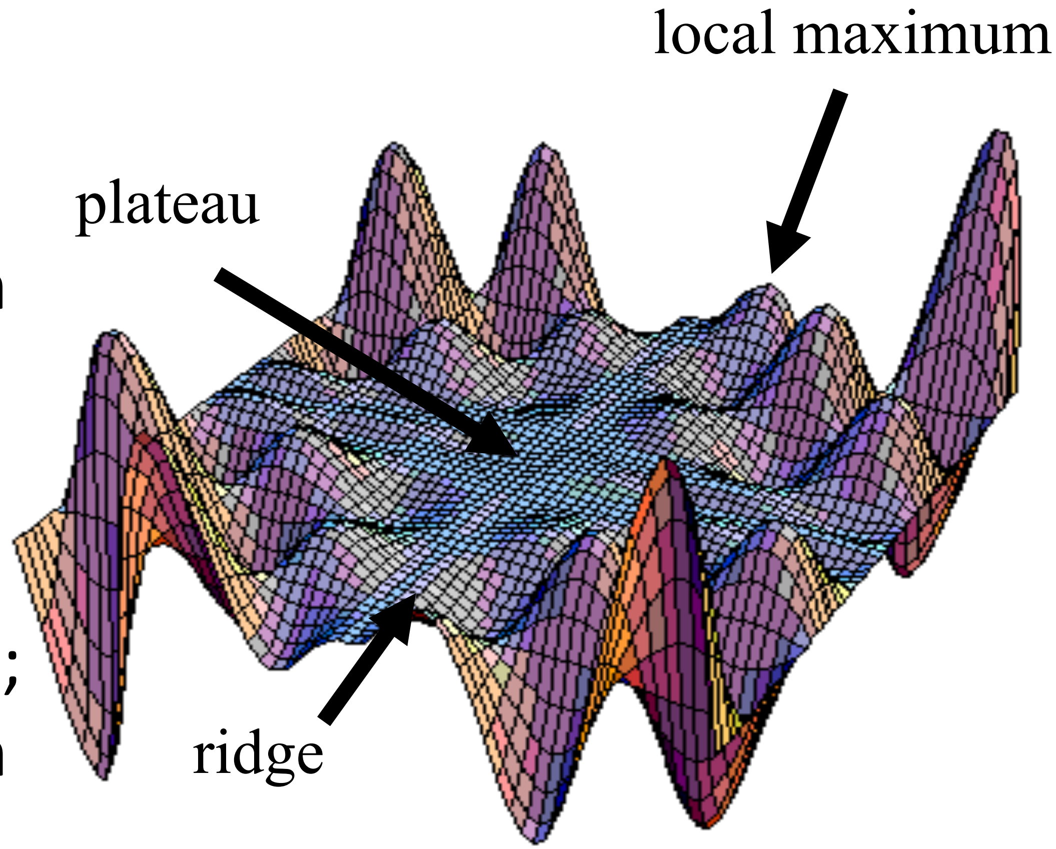


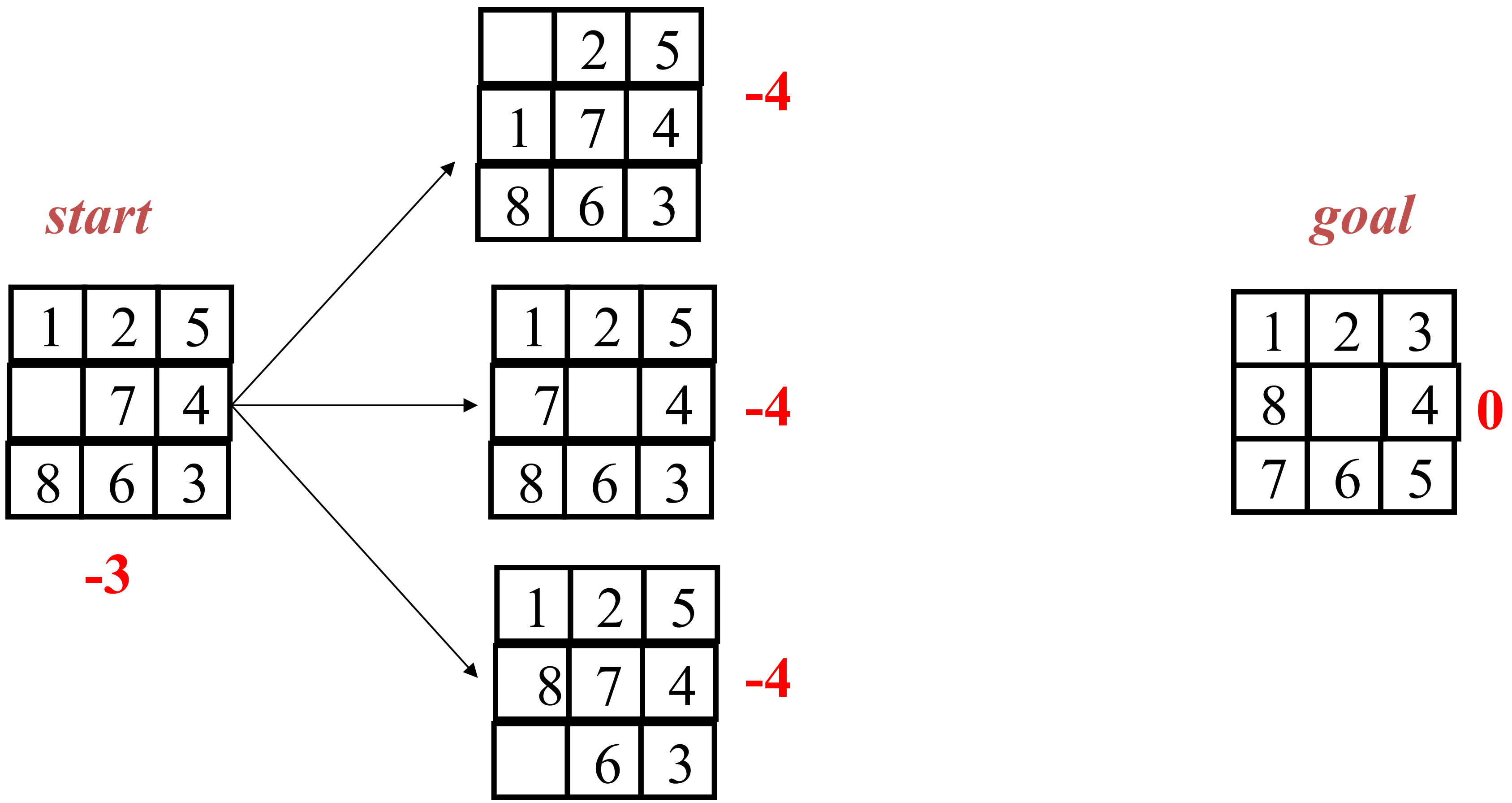
Image from: http://classes.yale.edu/fractals/CA/GA/Fitness/Fitness.html

Drawbacks of hill climbing

- Problems: local maxima, plateaus, ridges
- Possible remedies:
  - **Random restart:** keep restarting search from random locations until a goal is found

    may require an estimate – *how low can we go*
  - **Problem reformulation:** reformulate search space to eliminate these problematic features
- Some problem spaces are great for hill climbing and others are terrible

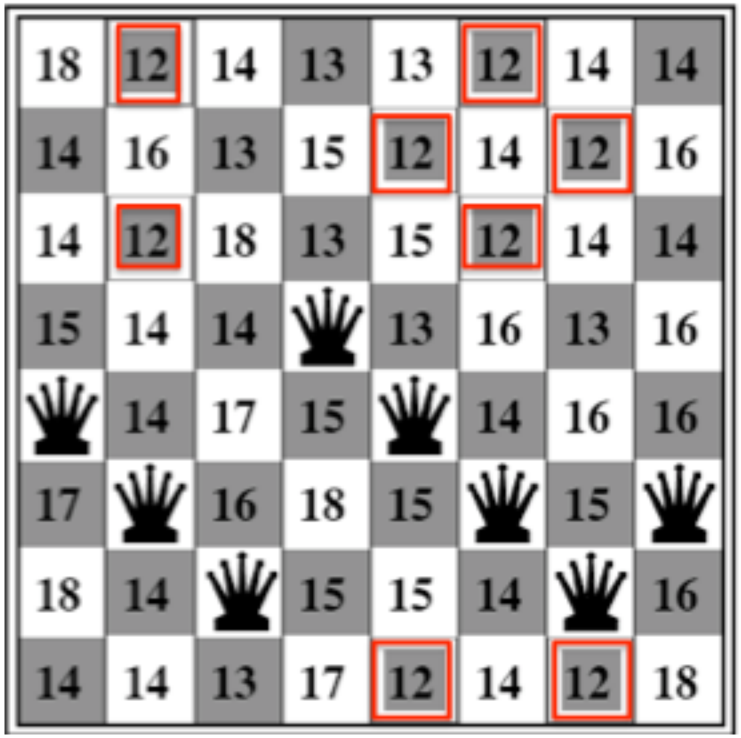
Example of a local optimum



Hill Climbing and 8 Queens

The 8 Queens problem often setup as follows:

- Randomly put one queen in each column
- Goal state: no two queens attack one another
- Actions: moving any queen to a different row
- Each state thus has 65 successors
- Heuristic h: # of pairs attacking one another
- Current state: h= 17
- h=0 => solution



Hill Climbing and 8 Queens

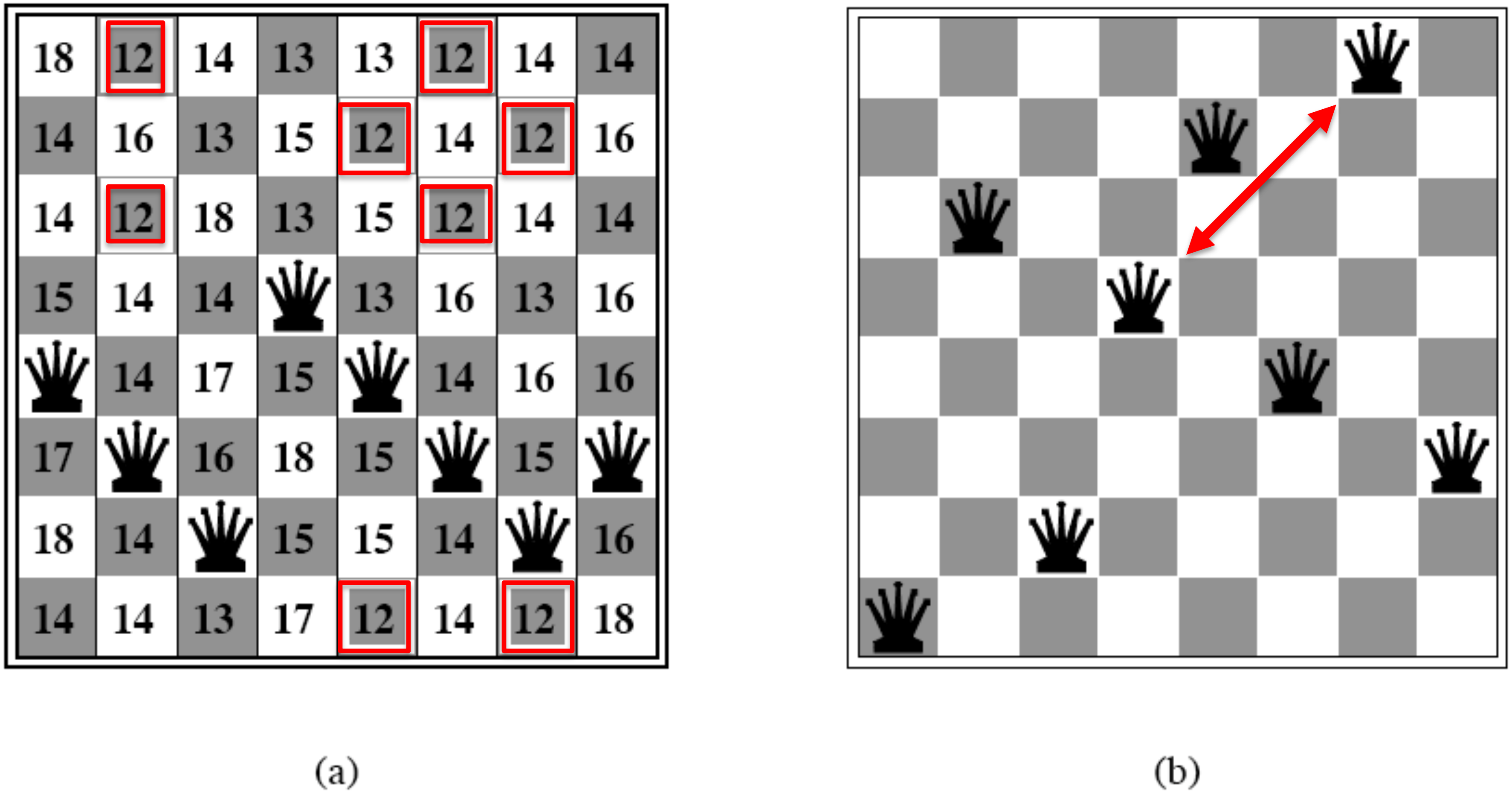


Figure 4.3 (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

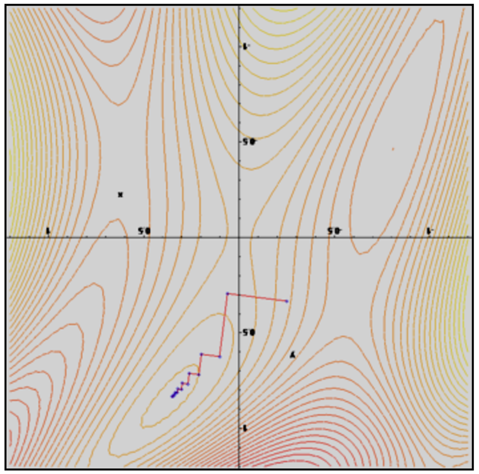
argmax and argmin

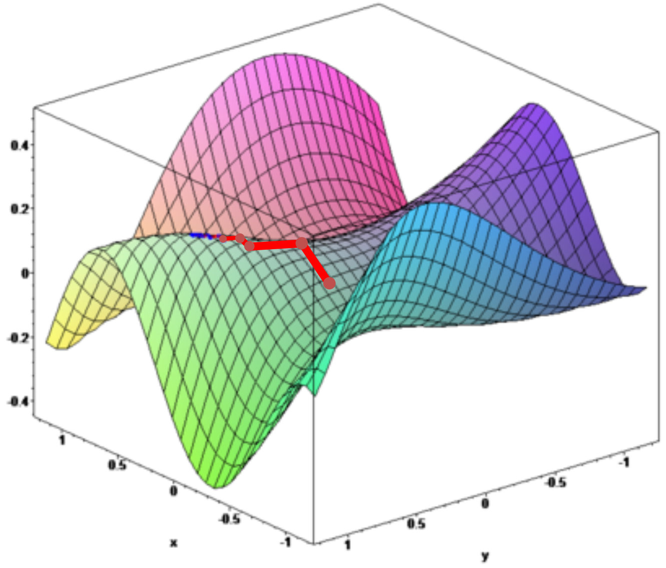
- The argmax and argmin concept is common in many AI and machine learning algorithms
- See argmax on Wikipedia (argmin is similar)

$$\underset{x}{\arg\max}\, f(x) := \{x \mid \forall y : f(y) \le f(x)\}.$$

- Argmax_x f(x) finds the value of x for which f(x) is largest

Gradient ascent or 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 is proportional to first derivative or slope;

  Steeper slope => bigger step
- Often used in machine learning algorithms
  - Cheaper to compute than Newton's method

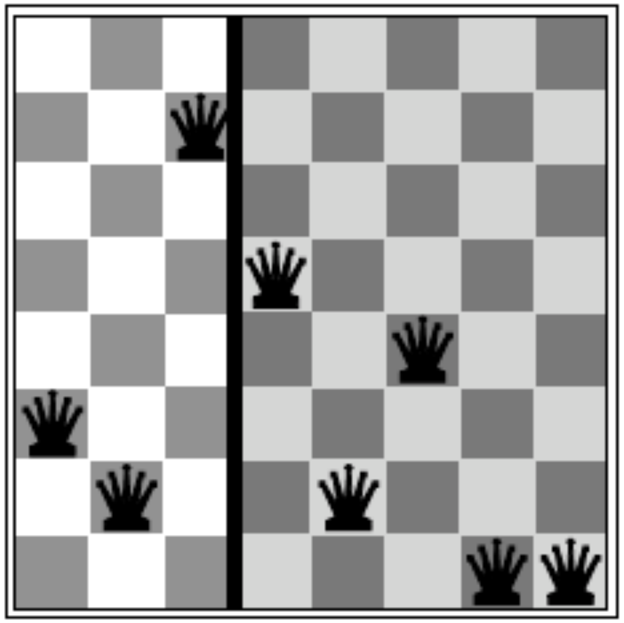
Local beam search

- 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)

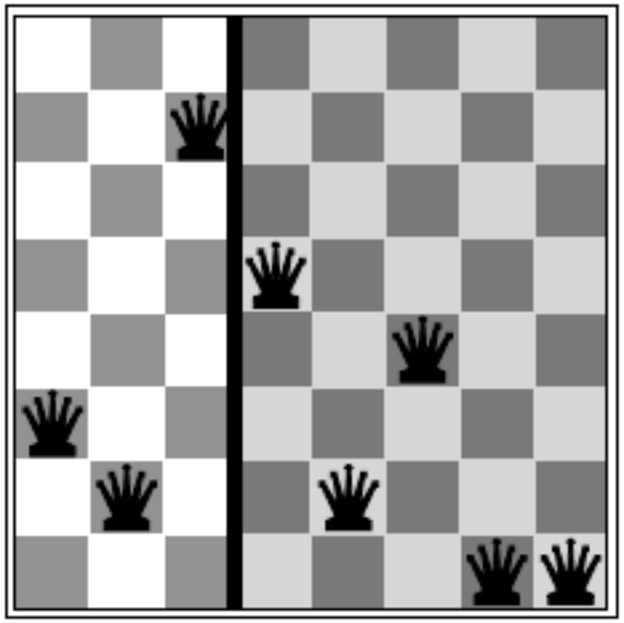
- Search technique inspired by *evolution*
- Similar to stochastic beam search
- Start with *initial population* of k random states
- New states generated by *mutating* a single state or *reproducing* (combining) two parent states, selected according to their *fitness*
- Encoding used for *genome* of an individual strongly affects the behavior of search

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
- $F(S_{solution}) = 8*7/2 = 28$
- $F(S_1) = 24$


State S_1

Genetic algorithms

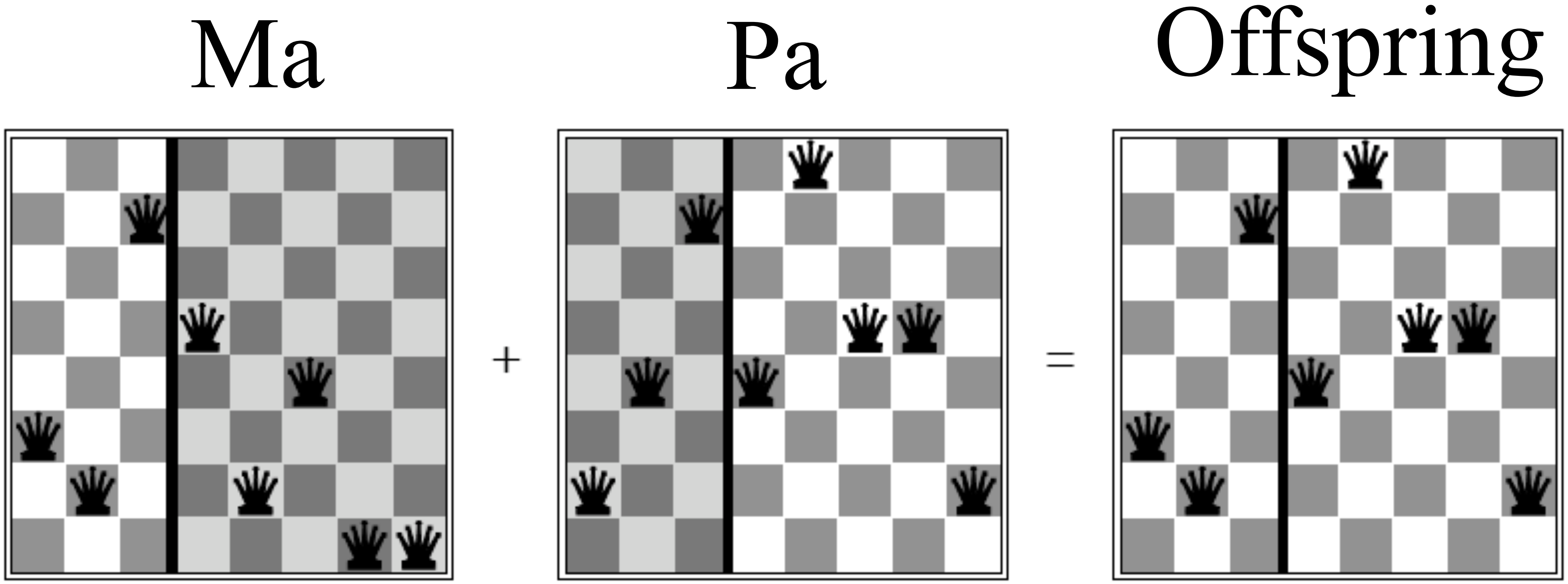


Figure 4.7 The 8-queens states corresponding to the first two parents in Figure 4.6(c) and the first offspring in Figure 4.6(d). The shaded columns are lost in the crossover step and the unshaded columns are retained.

Genetic algorithms

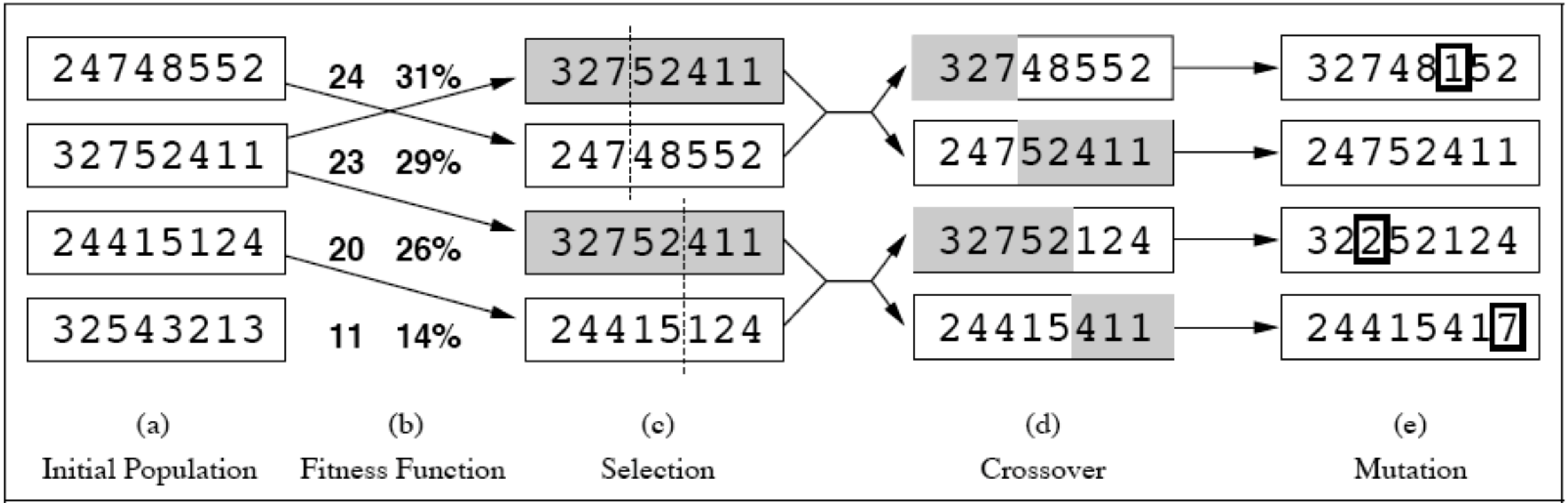


Figure 4.6 The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

- Fitness function: number of non-attacking pairs of queens (min=0, max=(8 × 7)/2 = 28)
- Probability of mating is a function of fitness score
- Cross-over point for a mating pair chosen randomly
- Resulting offspring subject to a random mutation with probability

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