Local Search Algorithms

Local Search Algorithms

- Local search algorithms operate by searching from a start state to neighbouring states, without keeping track of the paths, nor the set of states that have been reached.

- That means they are not systematic—they might never explore a portion of the search space where a solution actually resides.

- However, they have two key advantages:

(1) they use very little memory; and

(2) they can often find reasonable solutions in large or infinite state spaces for which systematic algorithms are unsuitable.

- Local search algorithms can also solve **optimization problems**, in which the aim is to find the best state according to an **objective function**.

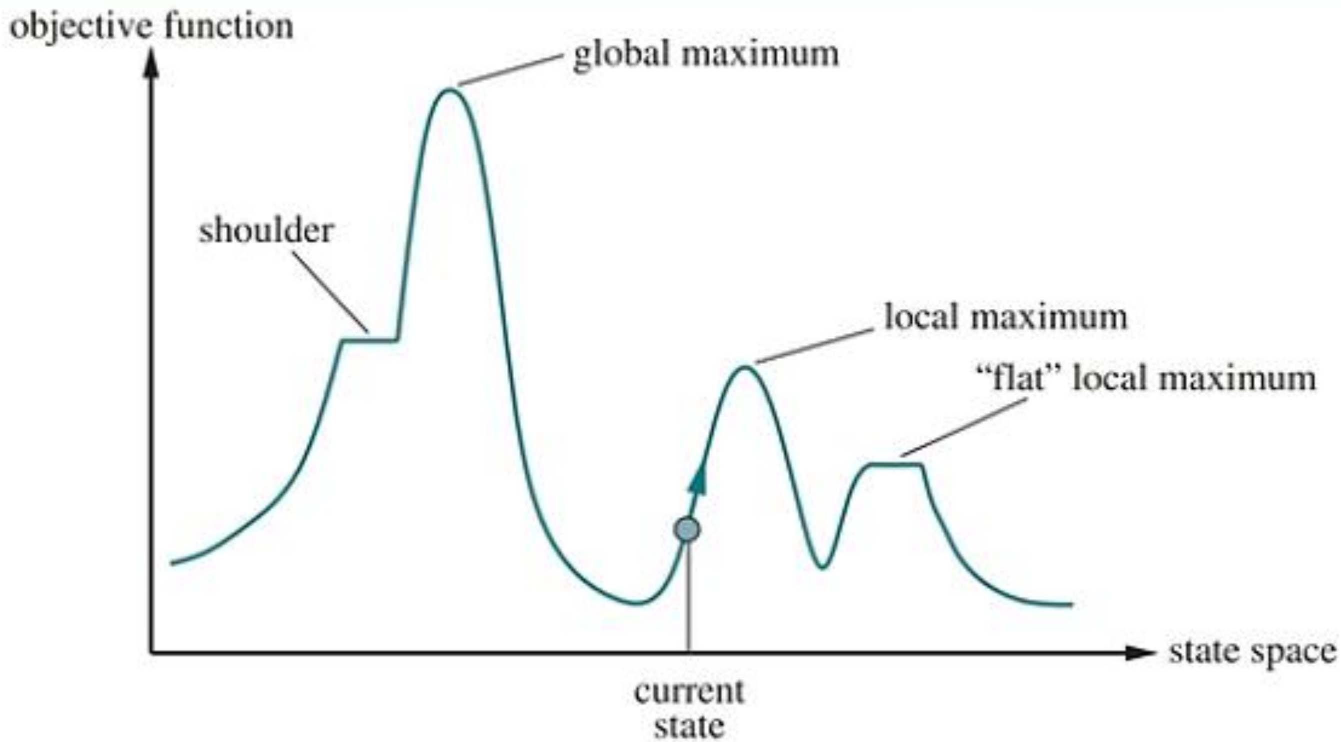


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum.

Hill-climbing search

- It keeps track of one current state and on each iteration moves to the neighbouring state with highest value
  - that is, it heads in the direction that provides the steepest ascent.
- It terminates when it reaches a “peak” where no neighbour has a higher value.

- Hill climbing does not look ahead beyond the immediate neighbours of the current state.

- Each point (state) in the landscape has an “elevation,” defined by the value of the objective function.

- If elevation corresponds to an objective function then the aim is to find the highest peak—a ***global maximum***—and we call the process **hill climbing**.

- If elevation corresponds to cost, then the aim is to find the lowest valley—a ***global minimum***—and we call it **gradient descent**.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← problem.INITIAL
  while true do
    neighbor ← a highest-valued successor state of current
    if VALUE(neighbor) ≤ VALUE(current) then return current
    current ← neighbor
```

Figure 4.2 The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor.

Hill Descent: Example (Minimizing h)

start

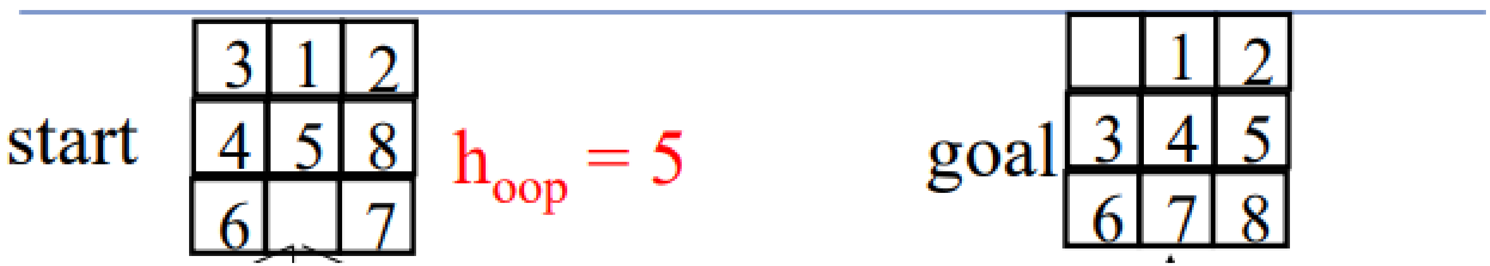
3	1	2
4	5	8
6		7

$h_{oop} = 5$

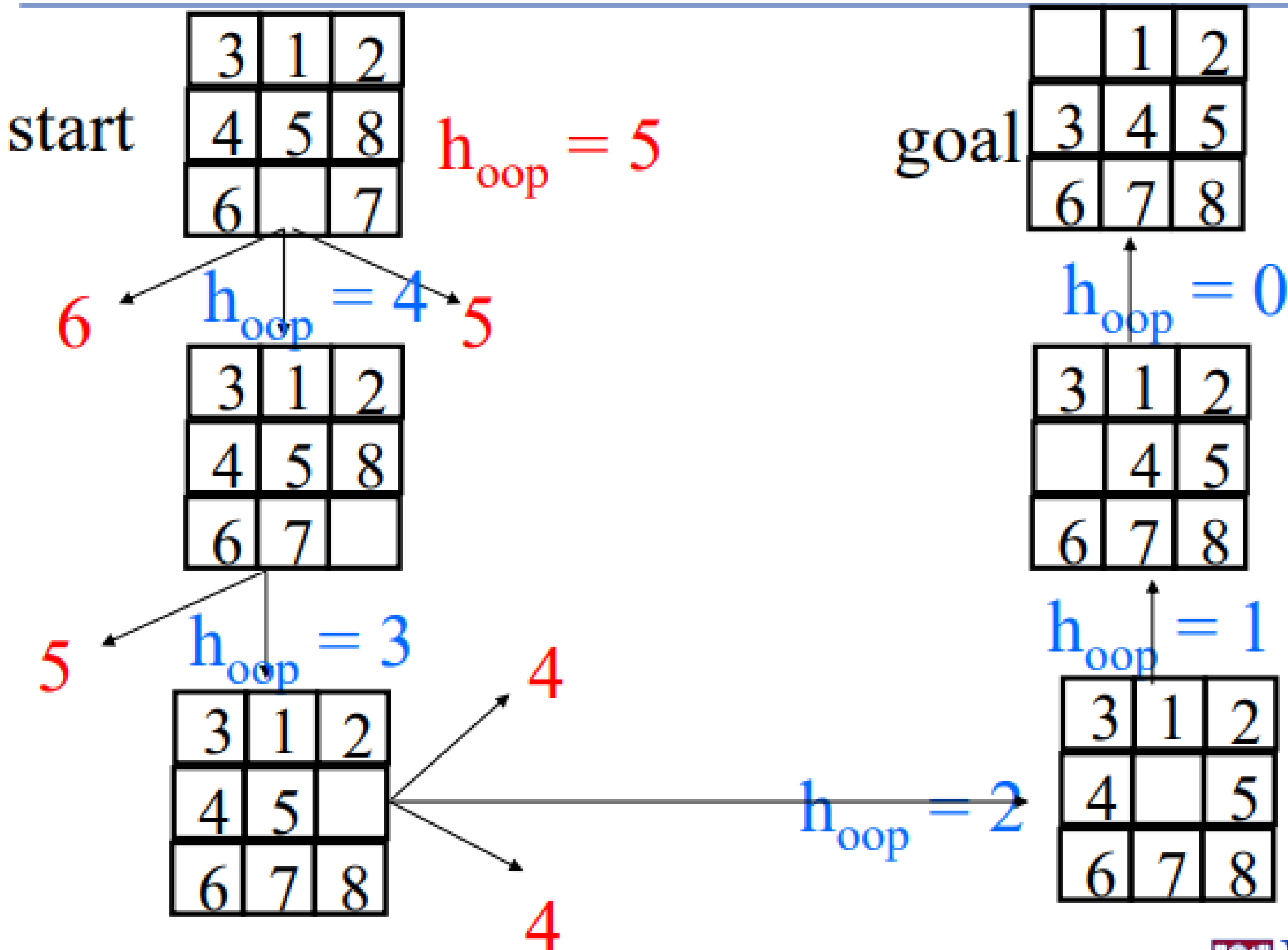
goal

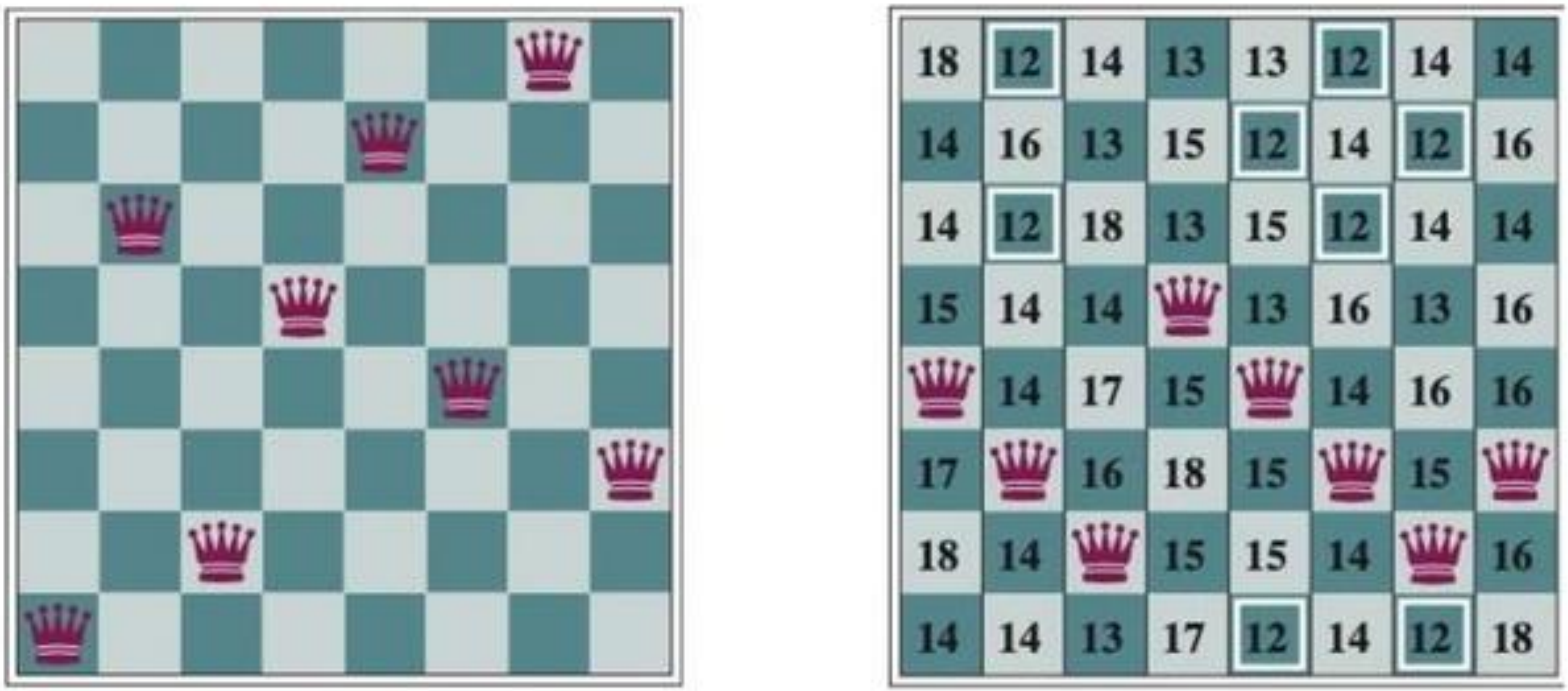
	1	2
3	4	5
6	7	8

Hill Descent: Example (Minimizing h)



Hill Descent: Example (Minimizing h)

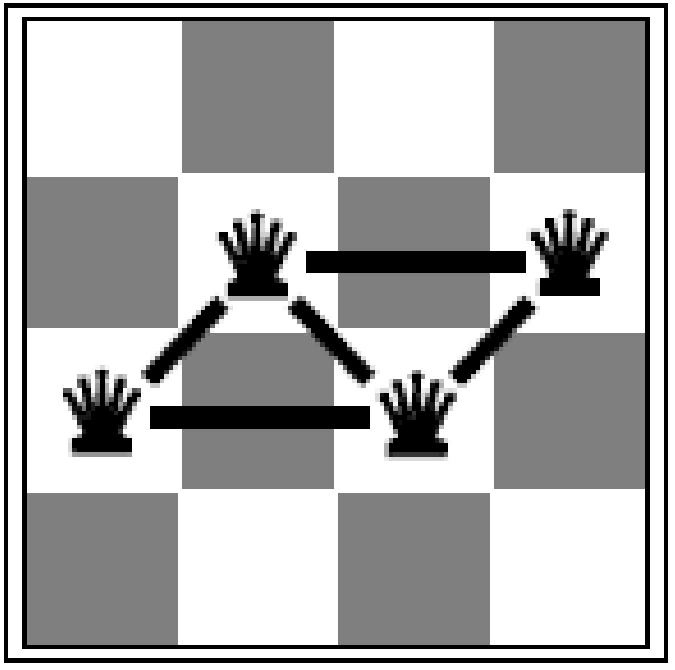




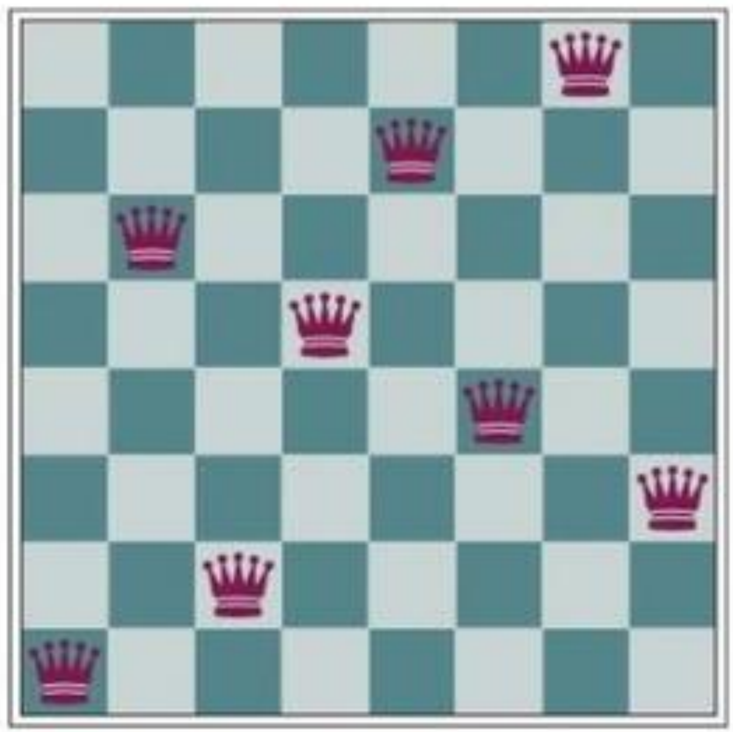
(a) (b)

Figure 4.3 (a) The 8-queens problem: place 8 queens on a chess board so that no queen attacks another. (A queen attacks any piece in the same row, column, or diagonal.) This position is almost a solution, except for the two queens in the fourth and seventh columns that attack each other along the diagonal. (b) An 8-queens state with heuristic cost estimate $h = 17$. The board shows the value of h for each possible successor obtained by moving a queen within its column. There are 8 moves that are tied for best, with $h = 12$. The hill-climbing algorithm will pick one of these.

Reduce the number of conflicts



h = 5



- Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next.

- Hill climbing can get stuck for any of the following reasons:

  - Local maxima: A local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum.

  - Ridges: Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate.

  - Plateaus: A plateau is a flat area of the state-space landscape

Types of Hill Climbing Algorithms

- **Stochastic hill climbing:** chooses at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move.

- **First-choice hill climbing:** implements stochastic hill climbing by generating successors randomly until one is generated that is better than the current state.

- **Random-restart hill climbing**: It conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found.

- A hill-climbing algorithm that never makes “downhill” moves toward states with lower value (or higher cost) is always vulnerable to getting stuck in a local maximum.

- In contrast, a purely random walk that moves to a successor state without concern for the value will eventually stumble upon the global maximum, but will be extremely inefficient.

- Combination of hill climbing with a random walk in a way that yields both efficiency and completeness.

Simulated annealing

- In metallurgy, annealing is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to reach a low-energy crystalline state.

- Simulated-annealing solution is to start by shaking hard (i.e., at a high temperature) and then gradually reduce the intensity of the shaking (i.e., lower the temperature).

- Instead of picking the best move, however, it picks a random move. If the move improves the situation, it is always accepted.
- Otherwise, the algorithm accepts the move with some probability less than 1.

- Basic idea:
  - Allow “bad” moves occasionally, depending on “temperature”
  - High temperature => more bad moves allowed, shake the system out of its local minimum
  - Gradually reduce temperature according to some schedule

Simulated annealing algorithm

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

current ← 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 $e^{\Delta E/T}$

Simulated Annealing

- The probability decreases exponentially with the “badness” of the move—the amount ΔE by which the evaluation is worsened.

- The probability also decreases as the “temperature” T goes down: “bad” moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases.

- If the schedule lowers T to 0 slowly enough, then a property of the Boltzmann distribution,

$$\boldsymbol{e^{\Delta E/T}}$$

- Is that all the probability is concentrated on the global maxima, which the algorithm will find with probability approaching 1.

Simulated Annealing

- Is this convergence an interesting guarantee?
- Sounds like magic, but reality is:
  - The more downhill steps you need to escape a local optimum, the less likely you are to ever make them all in a row
  - “Slowly enough” may mean exponentially slowly
  - Random restart hill climbing also converges to optimal state…
- Simulated annealing and its relatives are a key workhorse in VLSI layout and other optimal configuration problems

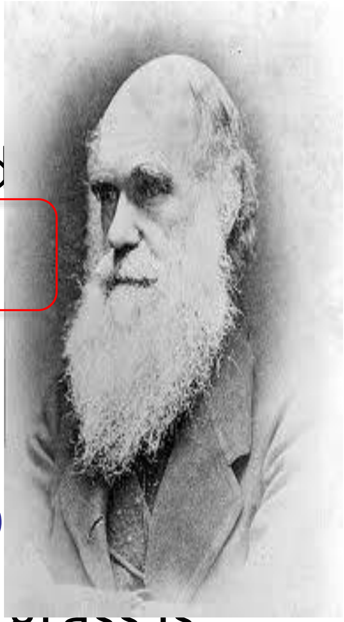
function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  current ← *problem*.INITIAL
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow$ VALUE(*current*) – VALUE(*next*)
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{-\Delta E/T}$

Figure 4.5 The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed. The *schedule* input determines the value of the "temperature" T as a function of time.

Local beam search

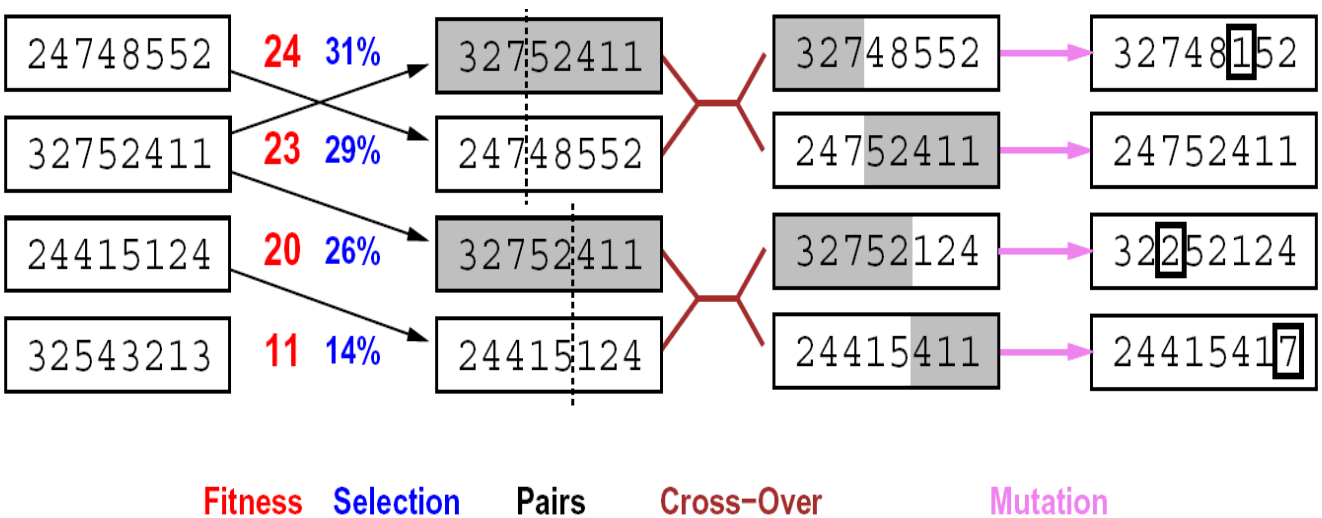
- Basic idea:
  - *K* copies of a local search algorithm, initialized rand
  - For each iteration
    - Generate ALL successors from *K* current states
    - Choose best *K* of these to be the new current states

Or, K chosen randomly with a bias towards good ones



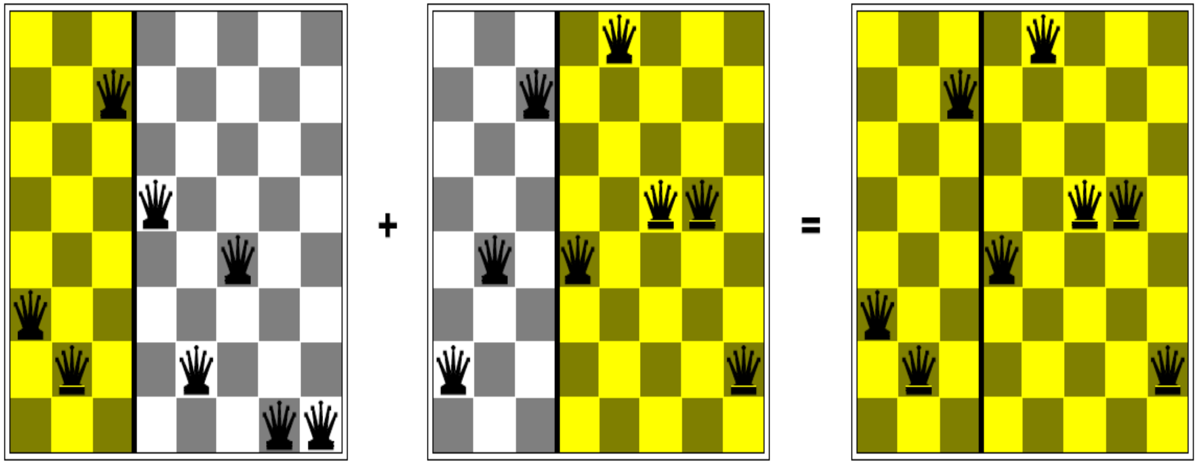
- Why is this different from *K* local searches in p
  - The searches ***communicate***! "Come over here, the grass is greener!"
- What other well-known algorithm does this remind you of?
  - Evolution!

Genetic algorithms



- Genetic algorithms use a natural selection metaphor
  - Resample *K* individuals at each step (selection) weighted by fitness function
  - Combine by pairwise crossover operators, plus mutation to give variety

Example: N-Queens



- Does crossover make sense here?
- What would mutation be?
- What would a good fitness function be?

Summary

- Many configuration and optimization problems can be formulated as local search
- General families of algorithms:
  - Hill-climbing, continuous optimization
  - Simulated annealing (and other stochastic methods)
  - Local beam search: multiple interaction searches
  - Genetic algorithms: break and recombine states

Many machine learning algorithms are local searches