CMSC 471
Fall 2012

Class #6

Tues 9/18/12
Local Search & Genetic Algorithms

Kevin Winner, winnerk1@cs.umbc.edu

Local Search and Genetic Algorithms

Sections 4.1, 4.2 and (4.5)

Outline

- Local Search
  - Hill Climbing
  - Gradient descent
  - Simulated Annealing
  - Local Beam Search
  - Genetic Algorithms!
  - Tabu Search
- Demo of HW2 domain

Iterative Improvement Search

- Another approach to search involves starting with an initial guess at a solution and gradually improving it until it is one.
- Work well in continuous domains
  – Optimization problems
    - 2D/3D Function Approximation
- Still work in graph-based domains
  – Work best if start state is not fixed
  – Reversible actions help too
- Some examples:
  – Hill Climbing
  – Simulated Annealing
  – Constraint satisfaction

Hill Climbing Search

- If there exists a successor s for the current state n such that
  - $h(s) < h(n)$
  - $h(s) \leq h(t)$ for all the successors t of n,
- then move from n to s. Otherwise, halt at n.
- Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.
- Similar to Greedy search in that it uses h, but does not allow backtracking or jumping to an alternative path since it doesn't "remember" where it has been.
- Corresponds to 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

start

2	8	3
1	6	4
7		5

h = -4

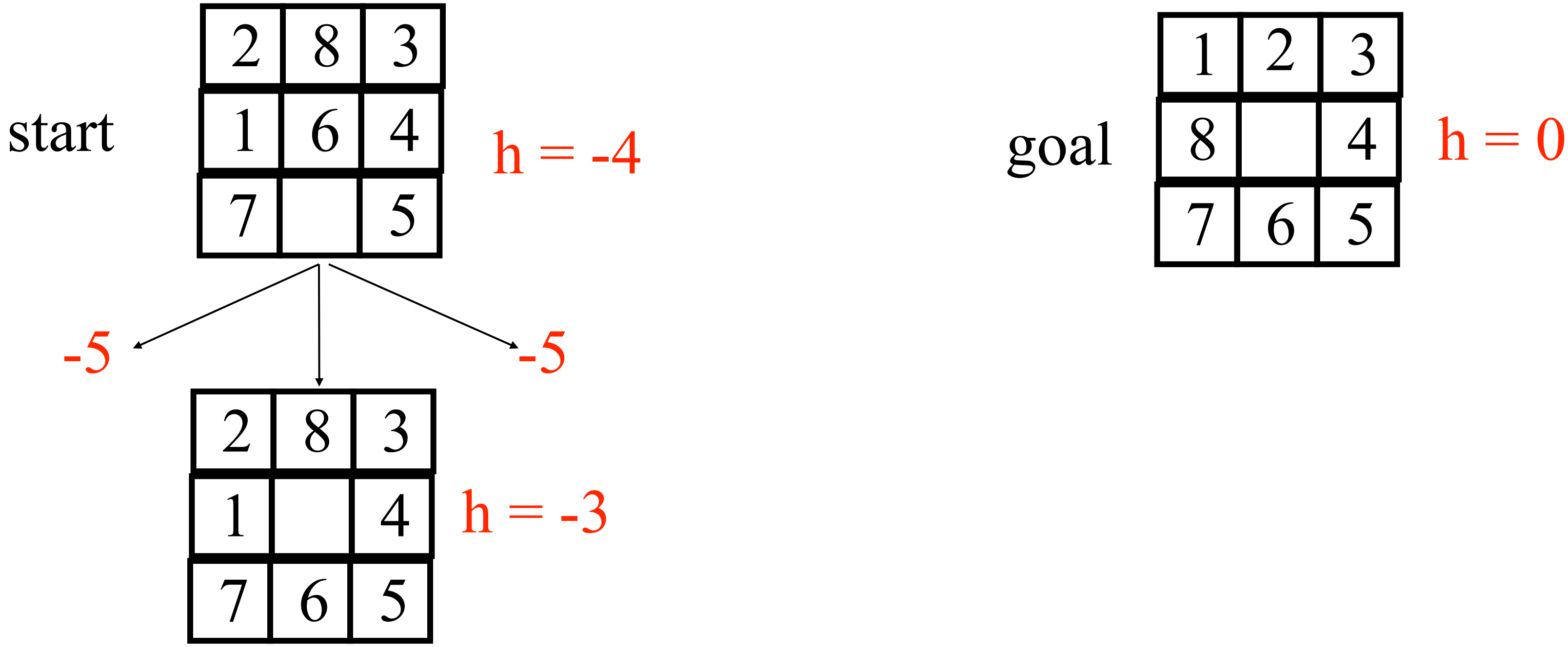
goal

1	2	3
8		4
7	6	5

h = 0

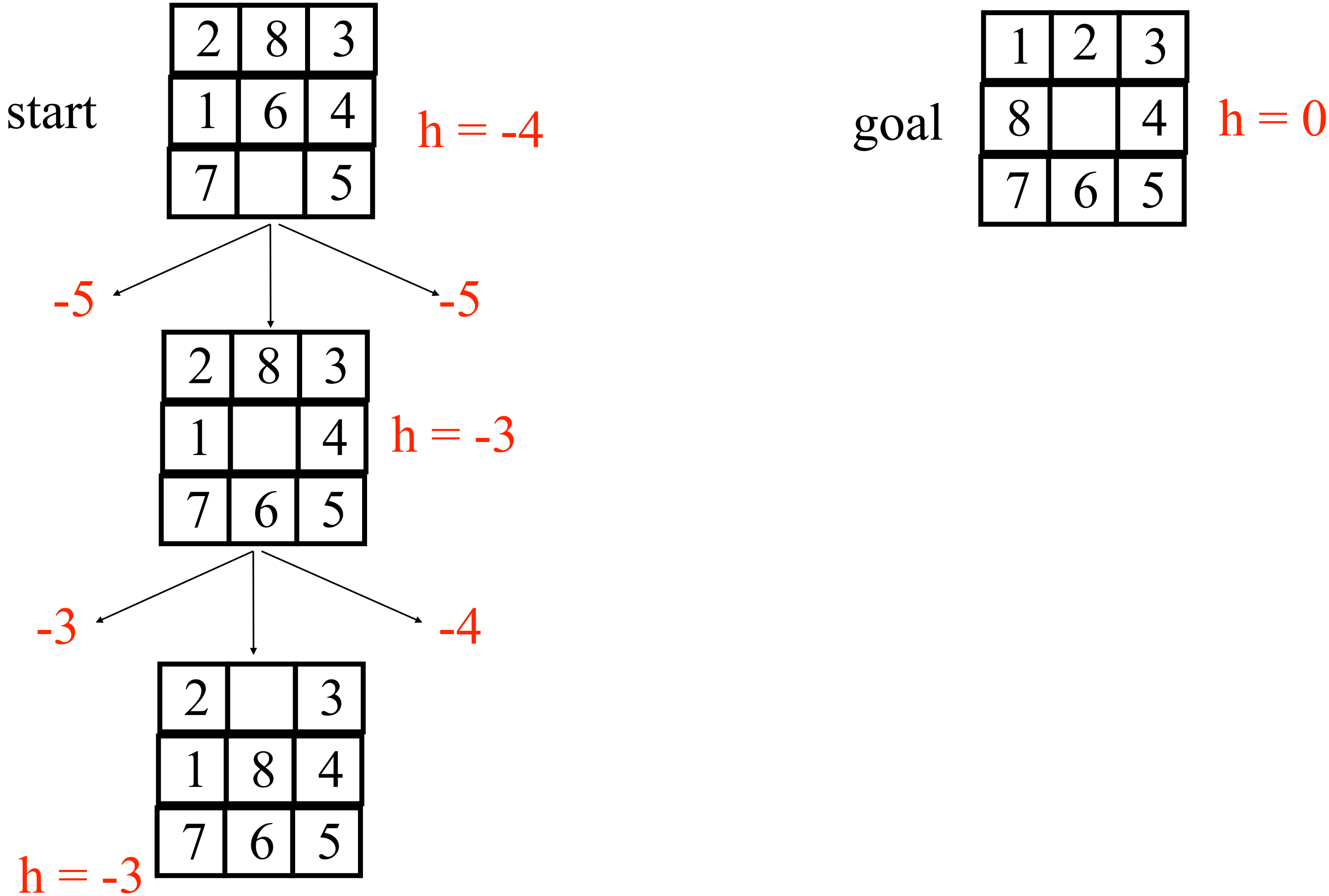
f(n) = -(number of tiles out of place)

Hill Climbing Example



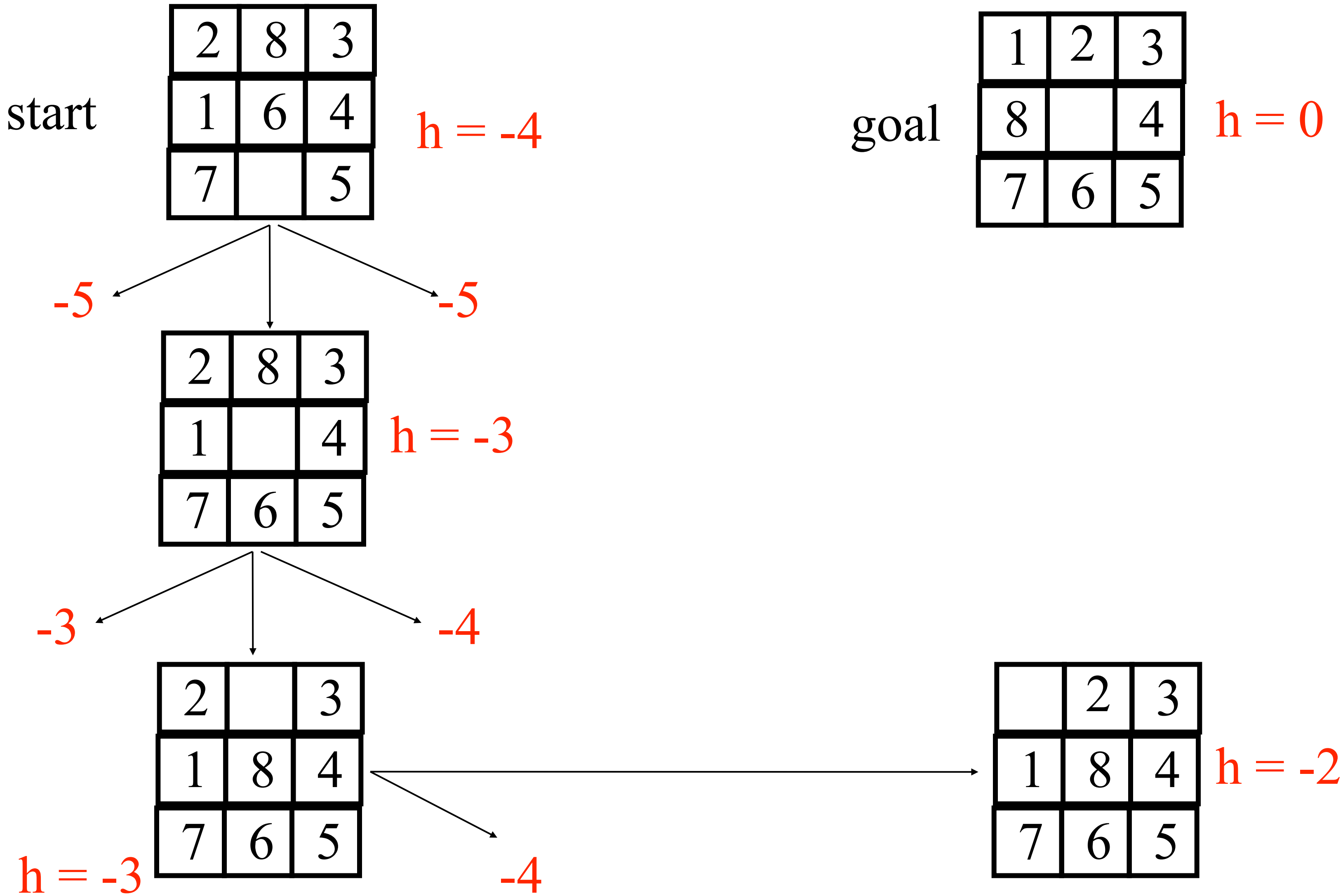
f(n) = -(number of tiles out of place)

Hill Climbing Example



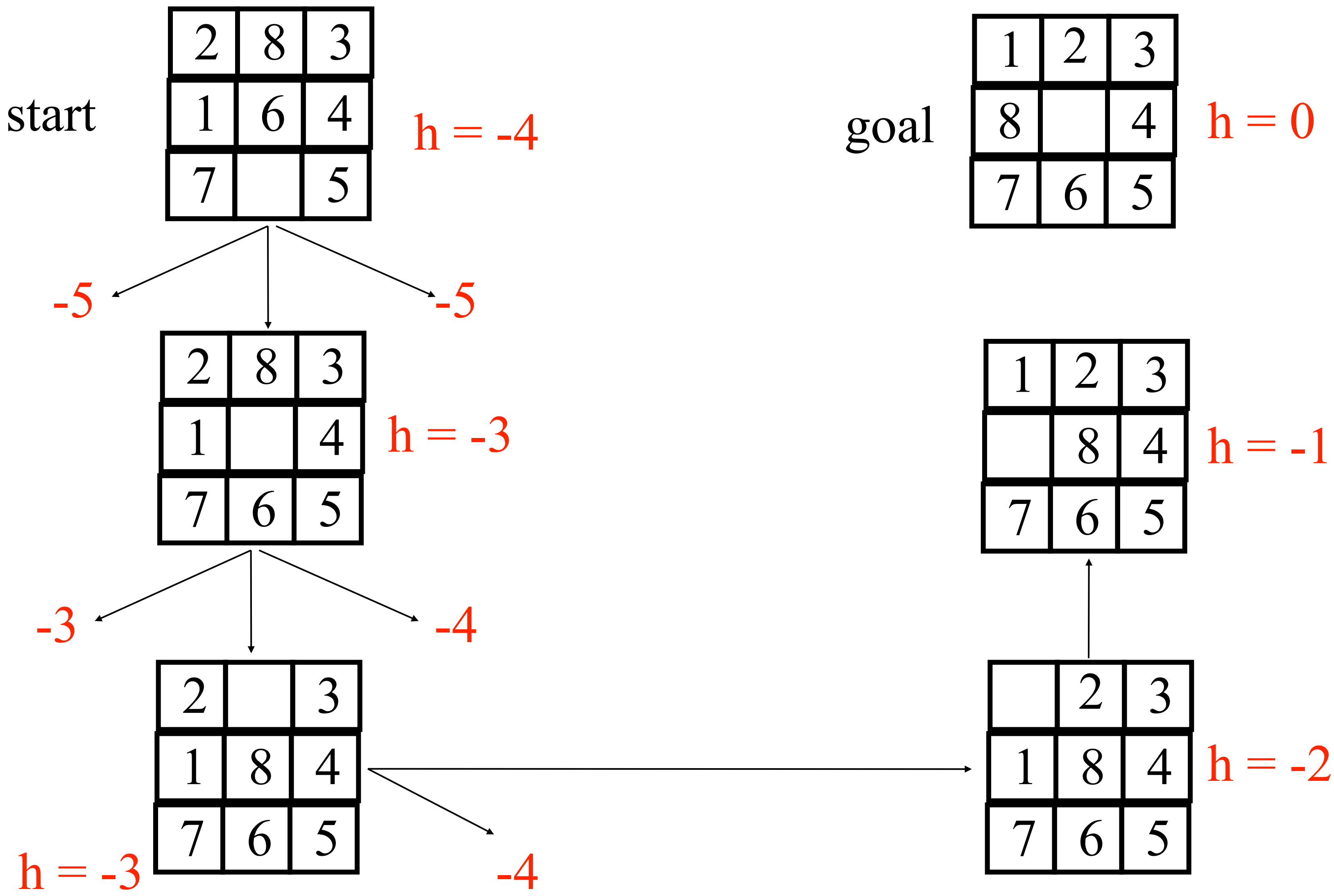
f(n) = -(number of tiles out of place)

Hill Climbing Example



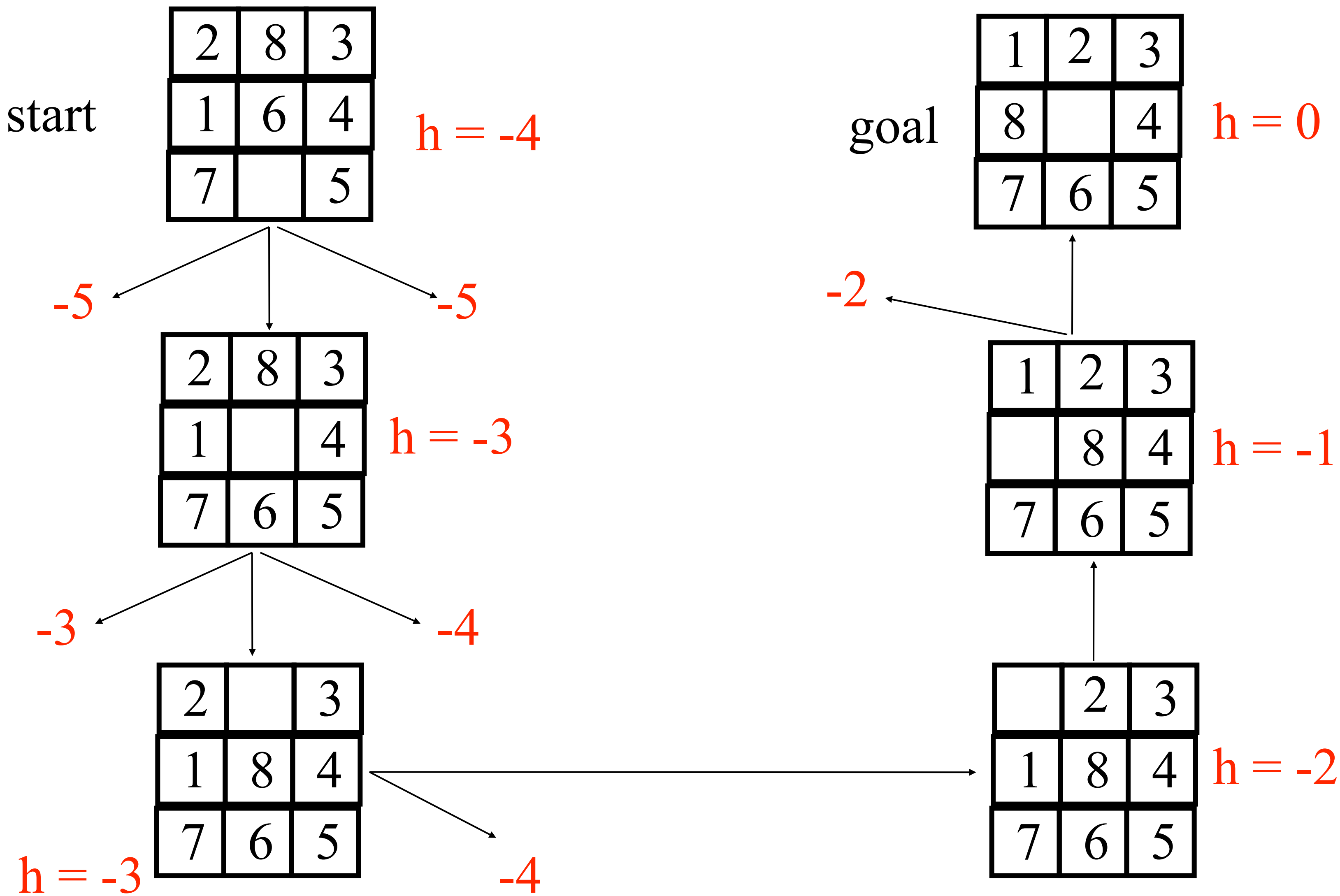
f(n) = -(number of tiles out of place)

Hill Climbing Example



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

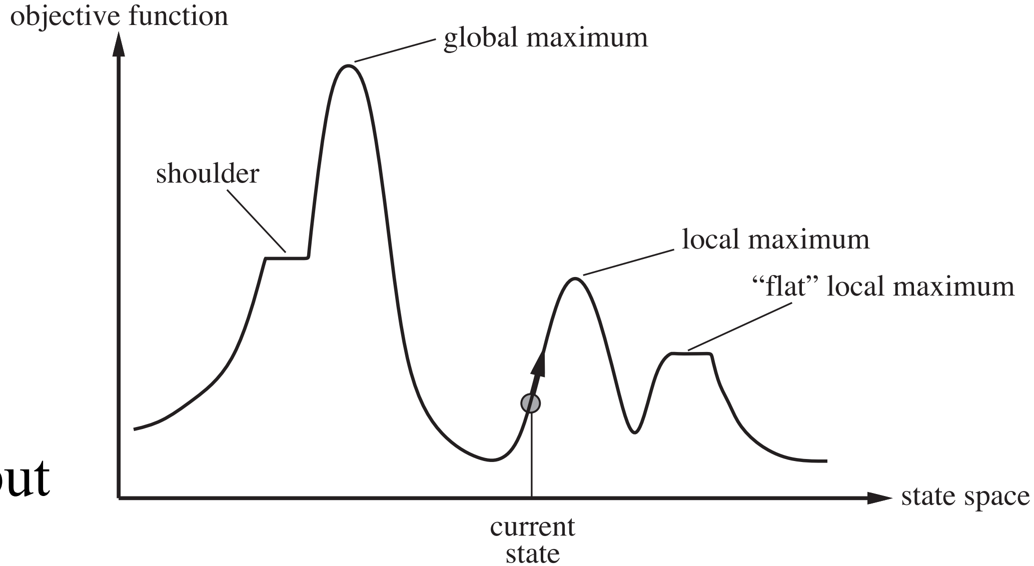
Hill Climbing Example



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

Exploring the Landscape

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



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

start

1	2	5
8	7	4
	6	3

f = -6

goal

1	2	3
8		4
7	6	5

f = 0

f = -(manhattan distance)

Example of a Local Optimum

start

1	2	5
8	7	4
	6	3

f = -6

move up

1	2	5
	7	4
8	6	3

f = -7

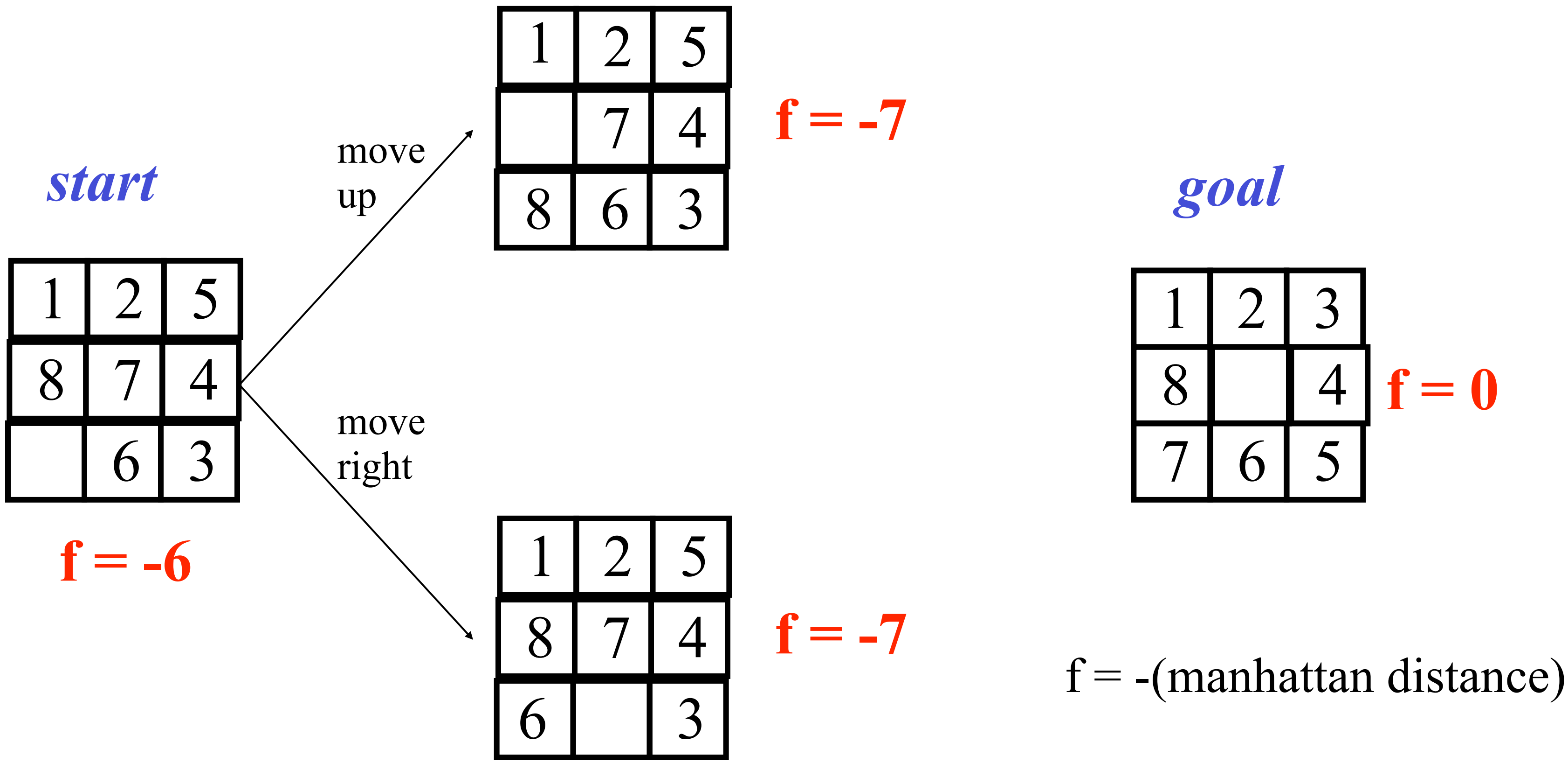
goal

1	2	3
8		4
7	6	5

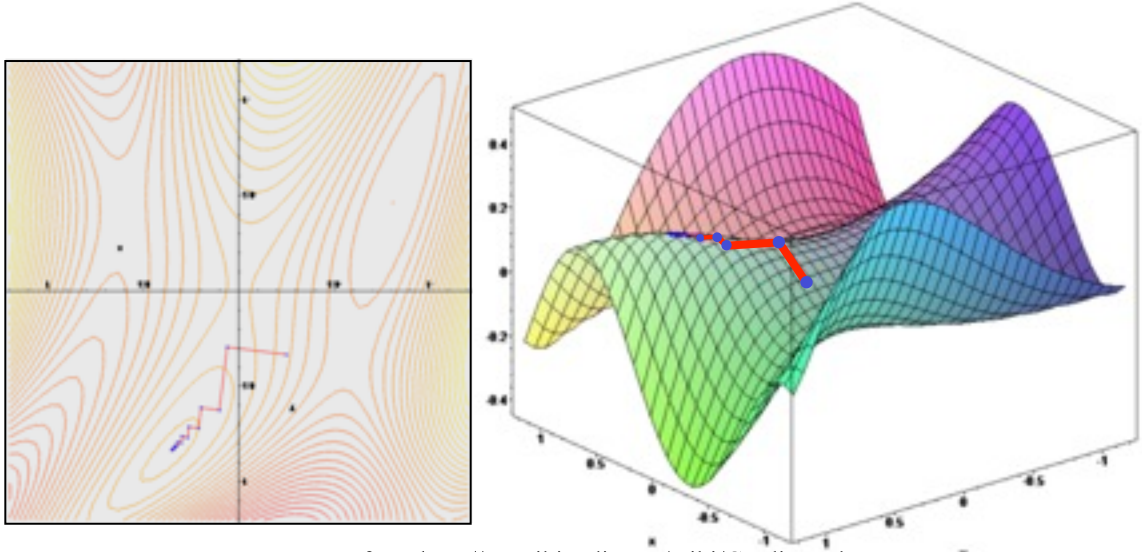
f = 0

f = -(manhattan distance)

Example of a Local Optimum



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
    - choose direction to walk
    - $x_{i+1} \leftarrow x_i + \eta \, \Delta f(x)$
  - until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
- Step size η is small (perhaps 0.1 or 0.05)
- Good for differentiable, continuous spaces

Simulated Annealing

- Simulated annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum-energy crystalline structure (the annealing process) and the search for a minimum [or maximum] in a more general system.
- SA can avoid becoming trapped at local minima.
- SA uses a random search that accepts changes that increase objective function f, **as well as** 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.

Simulated Annealing (cont.)

- f(s) represents the quality of state n (high is good)
- A “bad” move from A to B is accepted with a probability

$$P(\text{move}_{A \to B}) \approx e^{(f(B) - f(A)) \,/\, T}$$

  - (Note that f(b) – f(A) will be negative, so bad moves always have a relative probability less than one. Good moves, for which f(B) – f(A) is positive, have a relative probability greater than one.)
- 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

The Simulated Annealing Algorithm

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   static: current, a node
           next, a node
           T, a "temperature" controlling the probability 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^(ΔE/T)
```

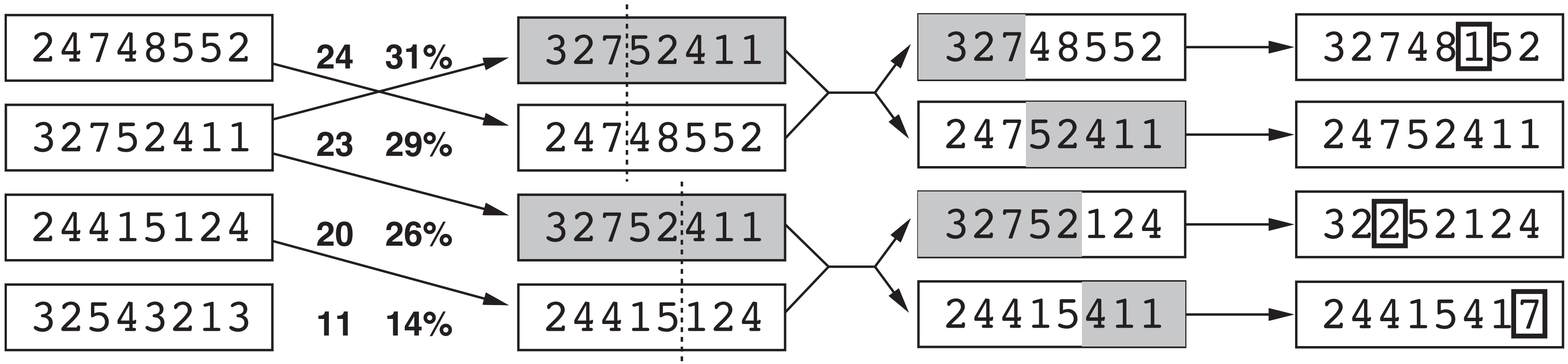
Local Beam Search

- Begin with k random states
- Generate all successors of these states
- Keep the k best states

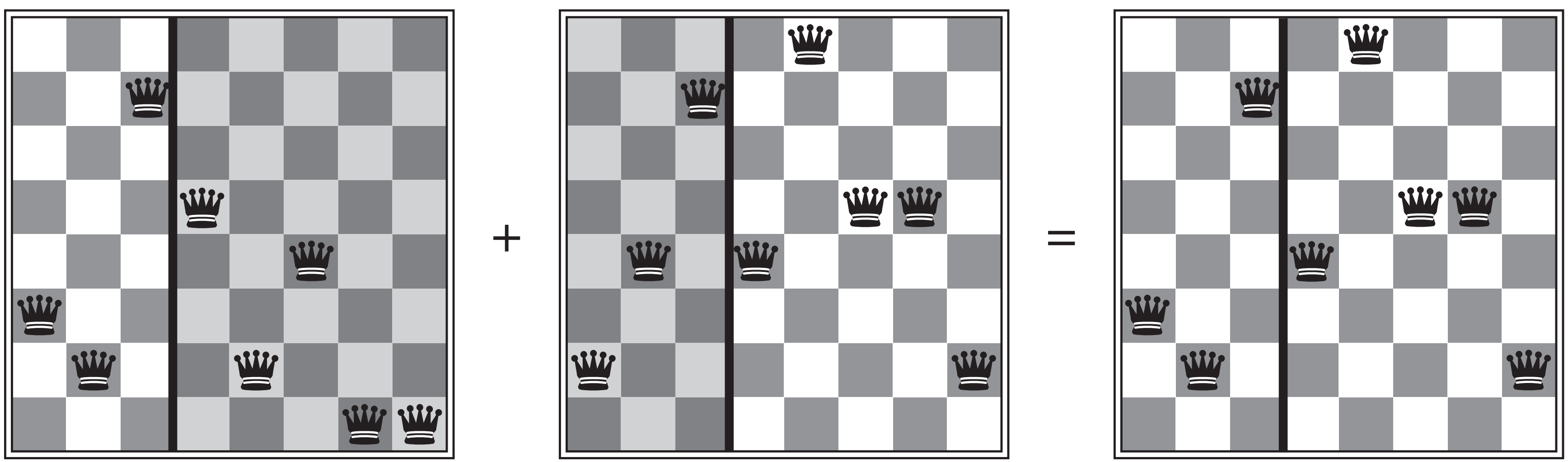
Genetic Algorithms

- Start with k random states (the *initial population)*
- New states are generated by "mutating" a single state or "reproducing" (combining via crossover) 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

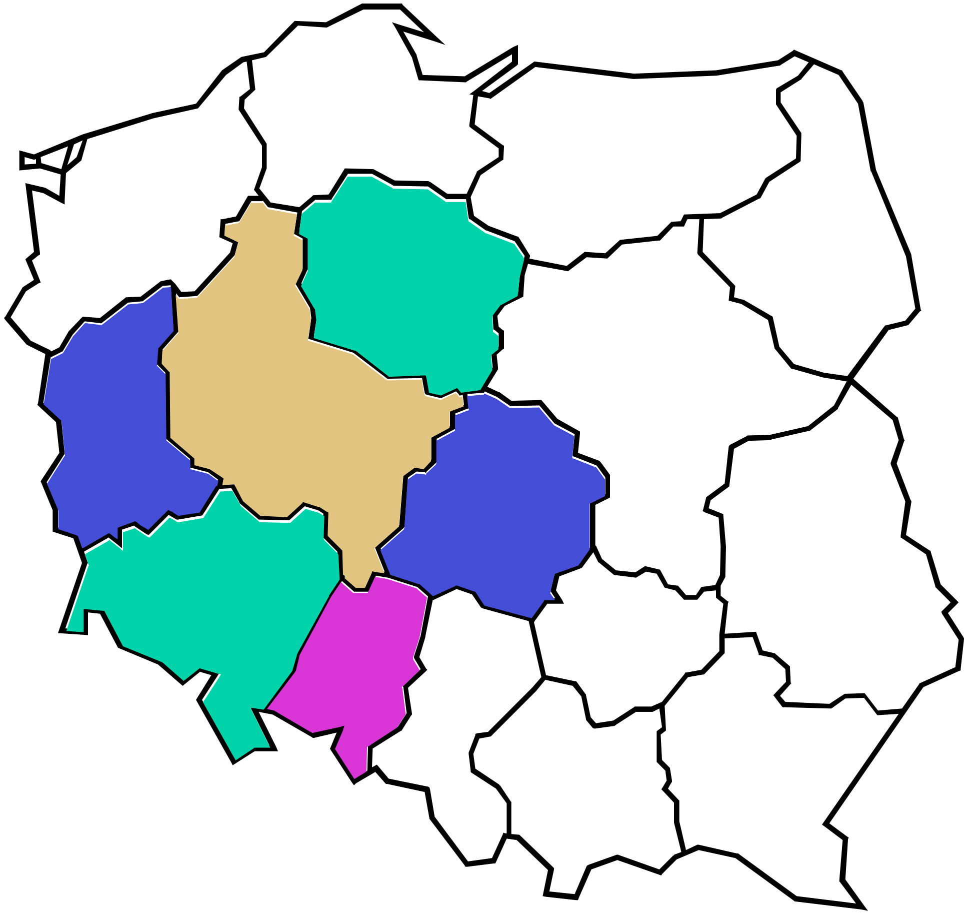


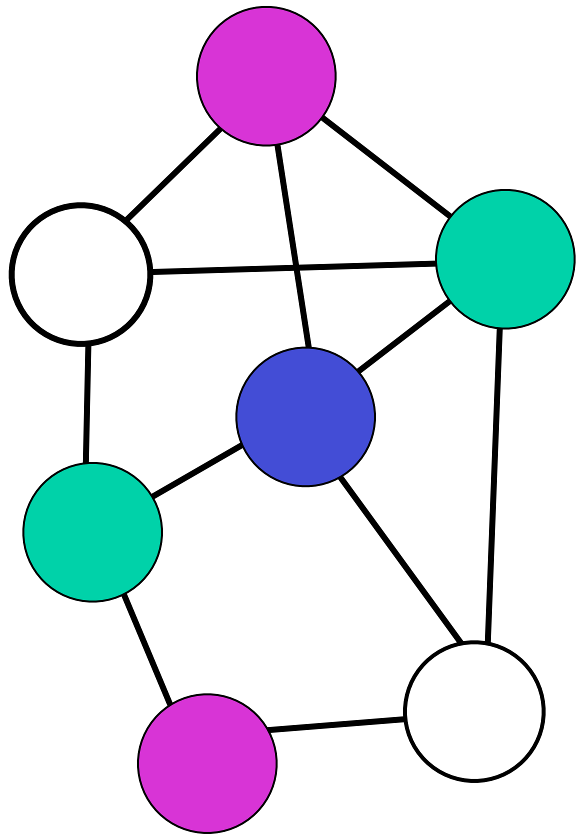
(a) Initial Population | (b) Fitness Function | (c) Selection | (d) Crossover | (e) Mutation



Class Exercise:

Local Search for Map/Graph Coloring





Class Exercise: Local Search for N-Queens

Q					
	Q				
		Q			
			Q		
				Q	
					Q

(more on constraint satisfaction heuristics next time...)

Summary: Informed Search

- **Best-first search** is general search where the minimum-cost nodes (according to some measure) are expanded first.
- **Greedy search** uses minimal estimated cost $h(n)$ to the goal state as measure. This reduces the search time, but the algorithm is neither complete nor optimal.
- **A* search** combines uniform-cost search and greedy search: $f(n) = g(n) + h(n)$. A* handles state repetitions and $h(n)$ never overestimates.
  – A* is complete and optimal, but space complexity is high.
  – The time complexity depends on the quality of the heuristic function.
  – IDA* and SMA* reduce the memory requirements of A*.
- **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.