Local Search
Ch. 4.1-4.2

Based on slides by Dr. Marie desJardin. Some material also adapted from slides by Dr. Matuszek @ Villanova University, which are based on Hwee Tou Ng at Berkeley, which are based on Russell at Berkeley. Some diagrams are based on AIMA.

1

Bookkeeping

- Upcoming: homework 1 due 9/19 at 11:59 PM (Monday)
- Guest lectures: 9/27 and 9/29
- Last time: informed (heuristic) search
  - Greedy search
  - A* and its variants
- Today:
  - Local search
  - Beginnings of constraint satisfaction?

2

Today's Class

- Local Search
  - Search as "landscape"
  - Iterative improvement methods
  - Hill climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
  - Online search
- Intro to Constraint Satisfaction

"If the **path** to the goal does not matter… [we can use] a single **current node** and move to neighbors of that node."

– R&N pg. 121

3

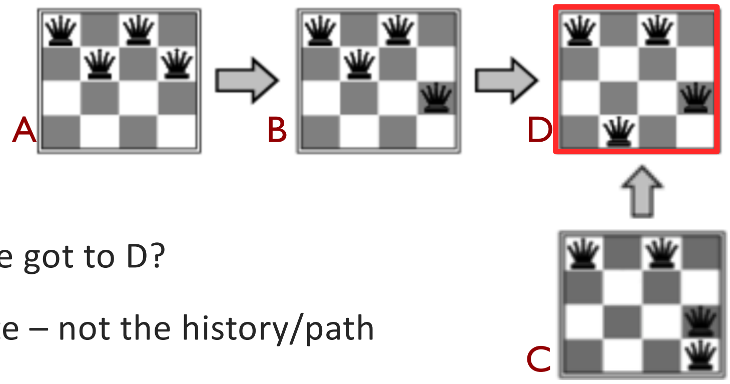
Local Search Algorithms

- Sometimes the path to the goal is irrelevant
  - Goal state itself is the solution
  - ∃ an **objective function** to evaluate states
- In such cases, we can use **local** search algorithms
- Keep a single "current" state, try to improve it

4

Local Search Example: n-Queens

- Put n queens on an n×n board with no two queens on the same row, column, or diagonal



- Does it matter how we got to D?
- We only need the state – not the history/path
- Once we reach D, can forget A, B/C

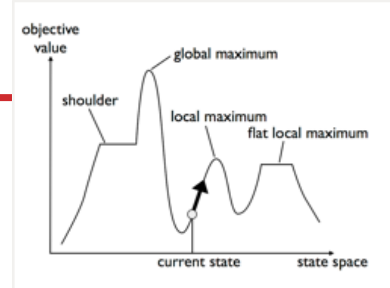
5

Local Search Algorithms

- Sometimes the path to the goal is irrelevant
  - Goal state itself is the solution
  - ∃ an **objective function** to evaluate states
- State space = set of "complete" configurations
  - That is, **all elements of a solution are present**
    - E.g., all the queens are on the board in some position
    - All sudoku squares are filled in
  - Find configuration satisfying constraints
- In such cases, we can use local search algorithms
- Keep a single "current" state, try to improve it

6

Landscapes



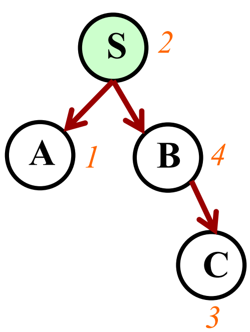
- Search graph can be a **landscape**
- Each node has **successor(s)** it can reach (called *s*)
  - Its children, unless there are loops
- Each successor has some "goodness" (desirability) according to the **objective** function
- $h(n) - h(s)$ is a positive, negative, or 0
- Want to go "uphill" (moving to a more desirable state)

Minor hassle:
Sometimes maximizing,
sometimes minimizing.

7

State Space (Landscape)

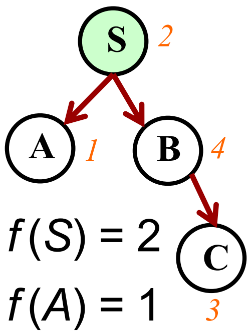
Maximizing (higher h(n) is better)



8

State Space (Landscape)

Maximizing (higher h(n) is better)



$f(S) = 2$

$f(A) = 1$

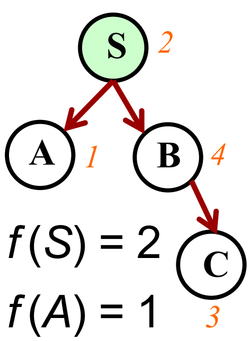
$f(B) = 4$

$f(C) = 3$

9

State Space (Landscape)

Maximizing (higher f(n) is better)

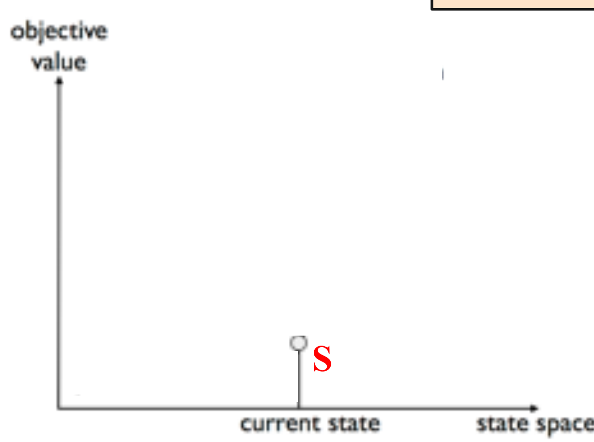


$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

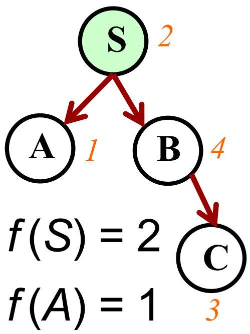
$f(C) = 3$



10

State Space (Landscape)

Maximizing (higher h(n) is better)

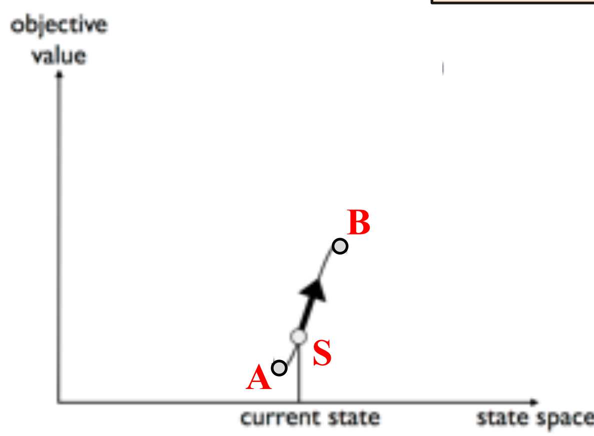


$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

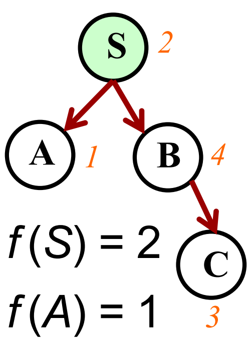
$f(C) = 3$



11

State Space (Landscape)

Maximizing (higher h(n) is better)

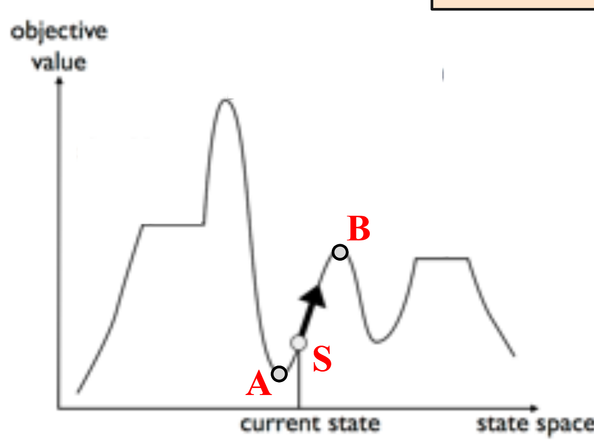


$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

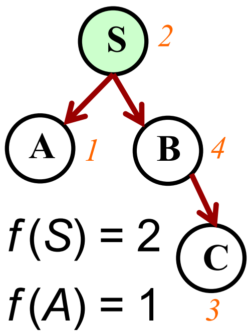
$f(C) = 3$



12

State Space (Landscape)

Maximizing (higher h(n) is better)

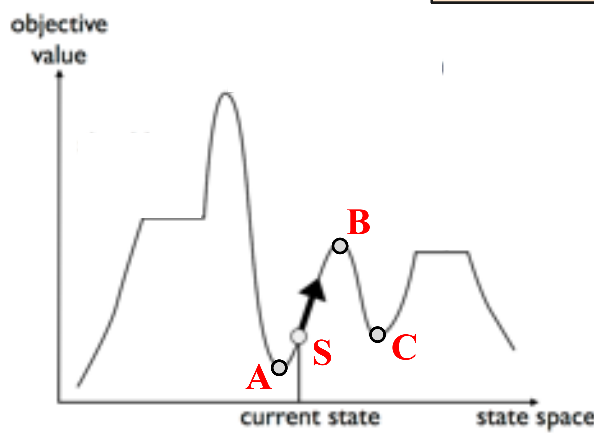


$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

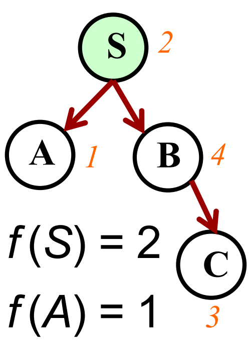
$f(C) = 3$



13

State Space (Landscape)

Maximizing (higher h(n) is better)

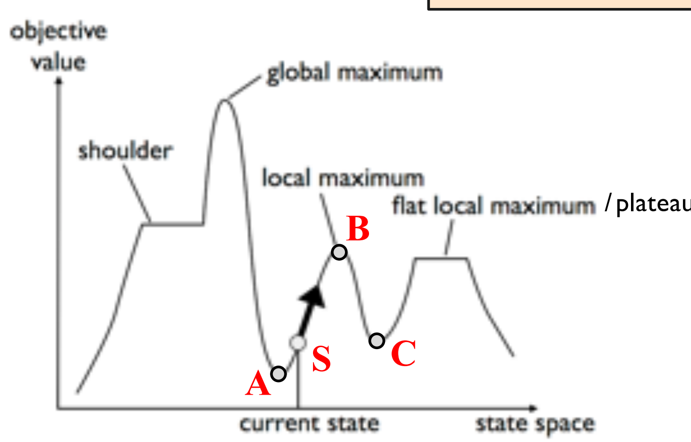


$f(S) = 2$

$f(A) = 1$

$f(B) = 4$

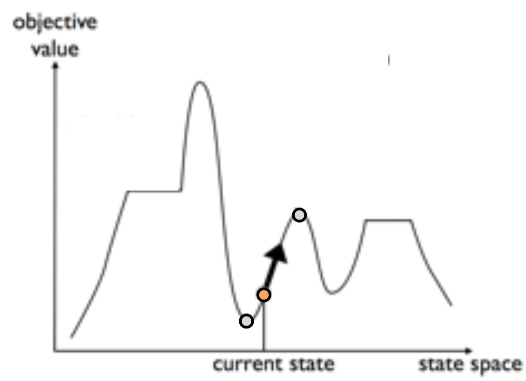
$f(C) = 3$



14

Iterative Improvement Search

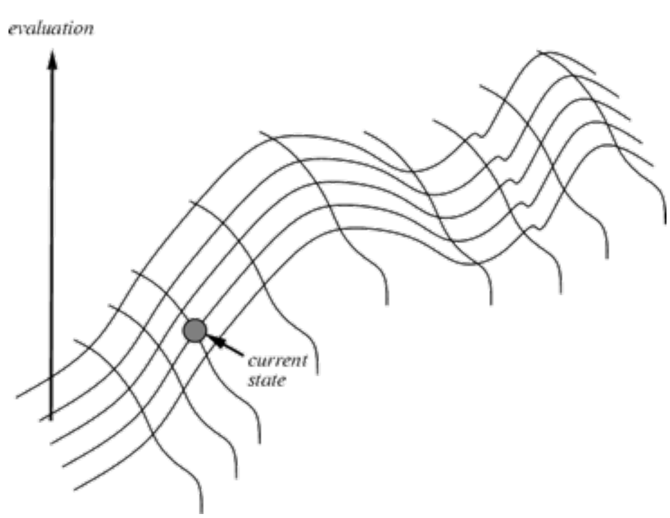
- Start with an initial guess
- Gradually improve it until it is legal or optimal
- Some examples:
  - Hill climbing
  - Simulated annealing
  - Constraint satisfaction



15

Hill Climbing on State Surface

- Concept: trying to reach the "highest" (most desirable) point (state)
- "Height" Defined by Evaluation Function
- Use the negative of heuristic cost function as the objective function



16

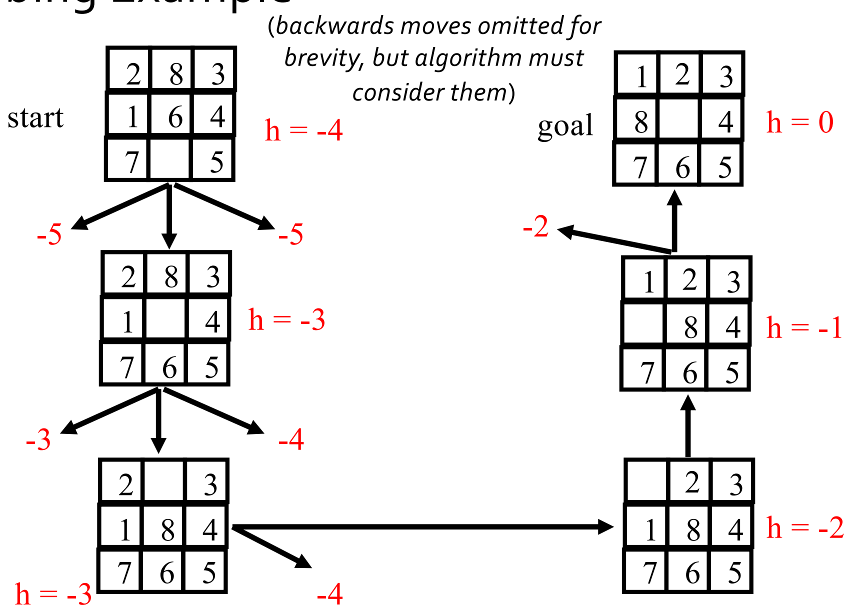
Hill Climbing Search

- Looks **one** step ahead to determine if any successor is "better" than current state, then moves to best choice
- If there exists a successor *s* for the current state *n* such that
  - $h(s) > h(n)$ – it's better than where we are now
  - $h(s) >= h(t)$ for all the successors *t* of *n* – and better than other choices

  then move from *n* to *s*. Otherwise, halt at *n*.
- A kind of Greedy search in that it uses *h*
  - But, does not allow backtracking or jumping to an alternative path
  - **Doesn't "remember" where it has been**
- Not *complete* or *optimal*
  - Search will terminate at local minima, plateaus, ridges.

17

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 whole space
- Plateaus:
  - A broad flat region that gives the search algorithm no direction (do a random walk)
- Ridges:
  - Flat like a plateau, but with drop-offs to the sides; steps to the North and South may go down, but a step to the East and West is stable

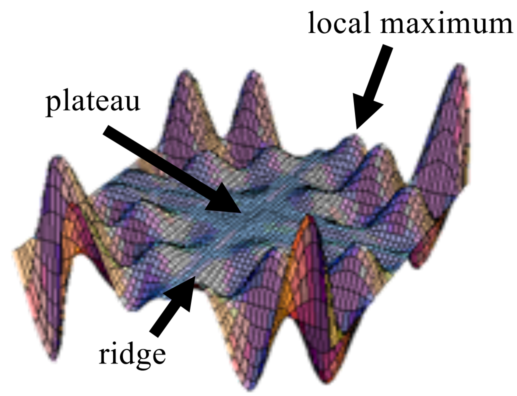


Image from: http://classes.yale.edu/fractals/CA/GA/Fitness/Fitness.html

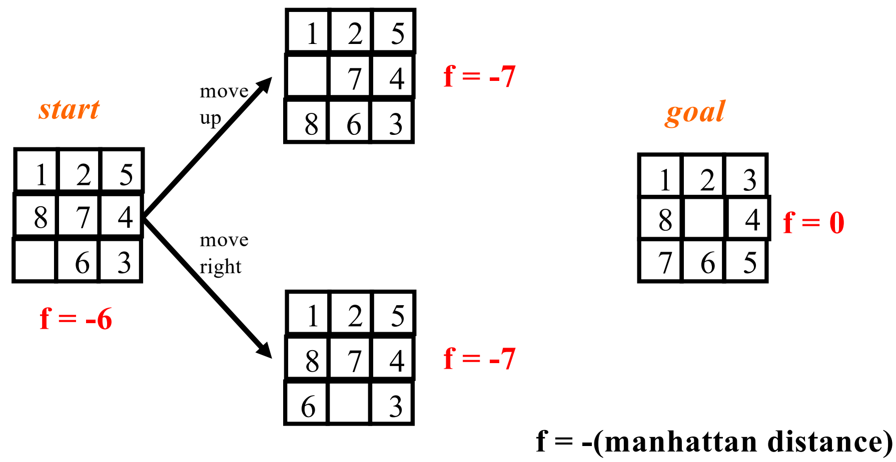
19

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; others are terrible

20

Example of a Local Optimum



21

Some Extensions of Hill Climbing

- Simulated Annealing
  - Escape local maxima by allowing some "bad" moves but gradually decreasing their frequency
- Local Beam Search
  - Keep track of *k* states rather than just one
  - At each iteration:
    - All successors of the *k* states are generated and evaluated
    - Best *k* are chosen for the next iteration

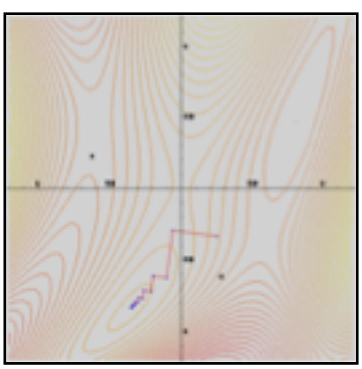
22

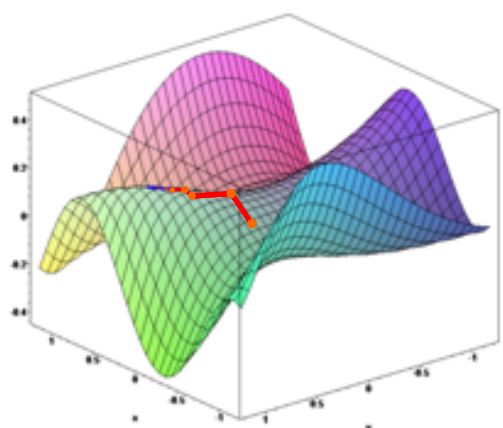
Some Extensions of Hill Climbing

- Stochastic (probabilistic) Beam Search
  - Chooses semi-randomly from “uphill” possibilities
  - “Steeper” (better) moves have a higher probability of being chosen
- Random-Restart Climbing
  - Can actually be applied to any form of search
  - Pick random starting points until one leads to a solution
- Genetic Algorithms
  - Each successor is generated from two predecessor (parent) states

23

Gradient Ascent / Descent





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

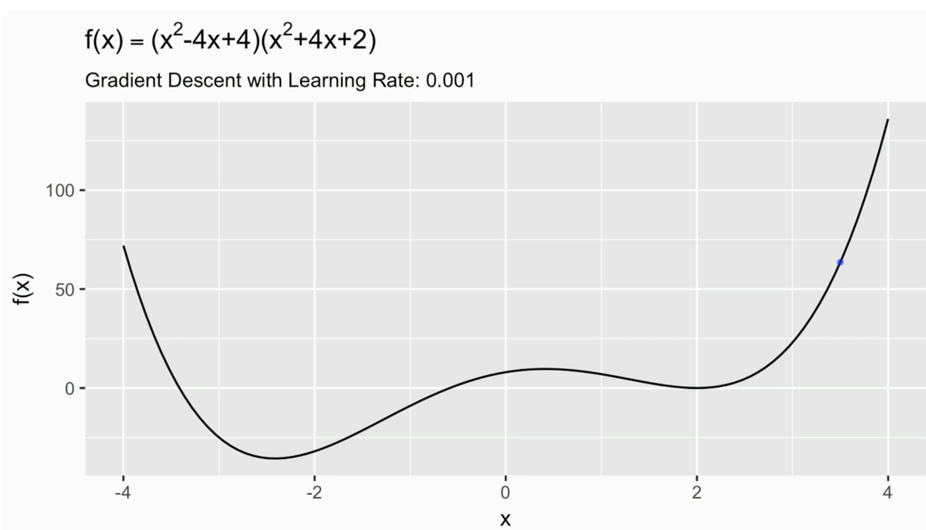
24

Gradient Descent (or Ascent)

- Length of downward "steps" proportional to negative of the gradient (slope) at the current state
  - "Steepest descent" → long "steps"
  - Jump to a node that is "farther away" if $f(\bullet)$ difference is large
- 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 (~0.1–0.05)
- Good for **differentiable, continuous** spaces

25

Gradient Descent

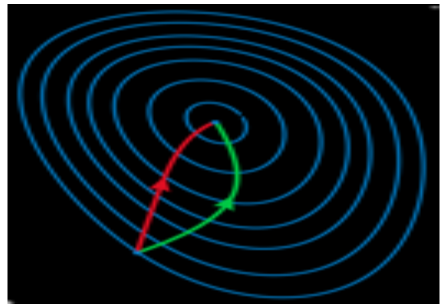


https://www.youtube.com/watch?v=CloIAJHZ3oE

26

Gradient Methods vs. Newton's Method

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



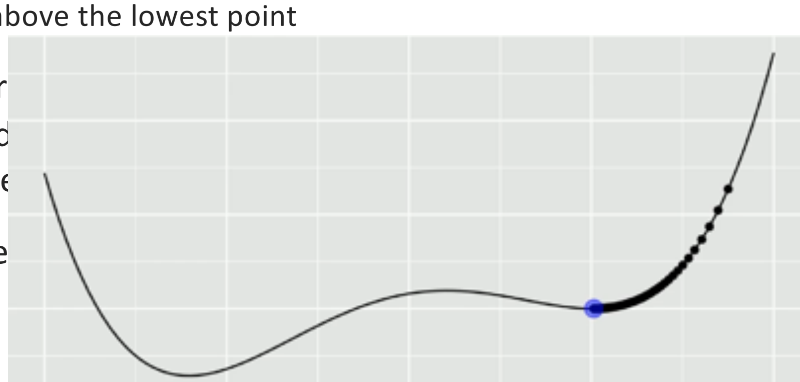
Contour lines of a function (blue)
• Gradient descent (green)
• Newton's method (red)

Images from http://en.wikipedia.org/wiki/Newton's_method_in_optimization

27

Simulated Annealing

- A hill-climbing algorithm that never makes "downhill" moves is vulnerable to getting stuck in a local maximum
  - For SA we'll consider local **minima** and reverse the objective function
  - Imagine a ball trying to reach the lowest state – it can get stuck in a "dip" that's above the lowest point
- A purely r … er it's "up" or "d … um, but is extreme…
- Therefore…



28

Simulated Annealing

- Conceptually: Escape **local maxima** by allowing some “bad” (locally counterproductive) moves but gradually decreasing their frequency
  - Our “ball” is allowed to bounce “up” occasionally
- Simulated annealing (SA): analogy between the way metal cools into a minimum-energy crystalline structure and the search for a minimum generally
  - In very hot metal, molecules can move fairly freely
  - They are slightly less likely to move out of a stable structure
  - As metal cools, molecules are more likely to stay

29

Simulated Annealing

- Can avoid becoming trapped at local minima.
- Uses a random local search that:
  - Accepts “moves” that decrease objective function f
  - **As well as some that increase it**
- Uses a control parameter T
  - By analogy with the original application
  - Is known as the system “temperature”

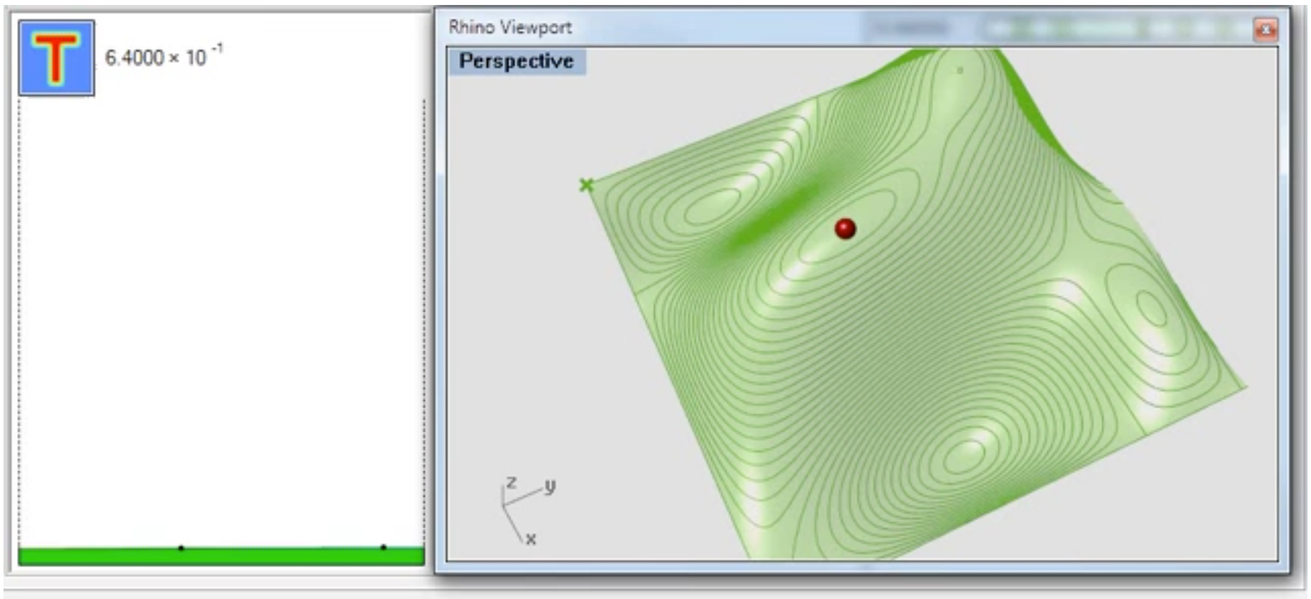
freedom to make “bad” moves

- T starts out high and gradually decreases toward 0

30

www.youtube.com/watch?v=VWtYLv-4oP0

Simulated Annealing: Examples



31

Simulated Annealing

- $f(n)$ represents the quality of state n (high is good)
- A "bad" move from A to B is accepted with probability

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

  - $f(B) - f(A)$ is negative – 'bad' moves have low probability
  - $f(B) - f(A)$ is positive – 'good' moves have higher probability
- Temperature
  - Higher temperature = more likely to make a "bad" move
  - As T tends to zero, this probability tends to zero
    - SA becomes more like hill climbing
  - If T is lowered slowly enough, SA is complete and admissible.

- domain-specific
- sometimes hard to determine

32

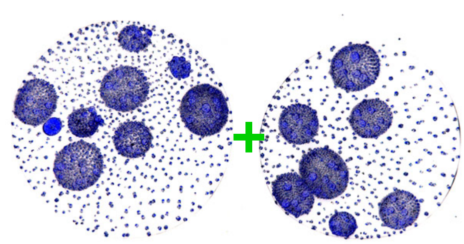
Local Beam Search

- Begin with *k* random states
  - *k*, instead of one, current state(s)
- Generate all successors of these states
- Keep the *k* best states across all successors
- Stochastic beam search
  - Probability of keeping a state is a function of its heuristic value
  - More likely to keep "better" successors

35

Genetic Algorithms

- The Idea:
  - New states generated by "mutating" a single state or "reproducing" (combining) two parent states
  - Selected for their **fitness**



- Similar to stochastic beam search
- Start with k random states (the **initial population**)
  - Encoding used for the "genome" of an individual strongly affects the behavior of the search
  - Must have some combinable representation of state spaces
  - Genetic algorithms / genetic programming are a research area

36

"Online" Search

- Interleave computation and action (search some, act some)
  - Exploration: Don't know outcomes of actions
  - So agent must try them!
- Competitive ratio = Path cost found* / Path cost that could be found**
  - * On average, or in an adversarial scenario (worst case)
  - ** If the agent knew transition functions and could use offline search
- Relatively easy if actions are reversible
- LRTA* (Learning Real-Time A*): Update h(s) (in a state table) as new nodes are found

More about online search and nondeterministic actions next time…

38

Summary: Local Search (I)

- State space can be treated as a "landscape" of movement through connected states
- We're trying to find "high" (good) points
- **Best-first search**: a class of search algorithms where minimum-cost nodes are expanded first
- **Greedy search**: uses minimal estimated cost h(n) to the goal state as measure of goodness
  - Reduces search time, but is neither complete nor optimal

39

Summary: Local Search (II)

- **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** search a space by modeling biological evolution
- **Online search algorithms** are useful in state spaces with partial/no information

Questions?

40

Class Exercise: Local Search for *n*-Queens

Q					
	Q				
		Q			
			Q		
				Q	
					Q

Heuristic?
State space?
Search algorithm?
Example moves?
Problems?

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

41