CS 1571 Introduction to AI
Lecture 11

Finding optimal configurations II

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Announcements

- **Homework assignment 3 due today**
- **Homework assignment 4 is out**
  - Programming and experiments
  - Simulated annealing + Genetic algorithm
  - Competition

Course web page:

http://www.cs.pitt.edu/~milos/courses/cs1571/

Search for the optimal configuration

Objective:

- **find the optimal configuration**

Optimality:

- Is efined by some **quality measure**

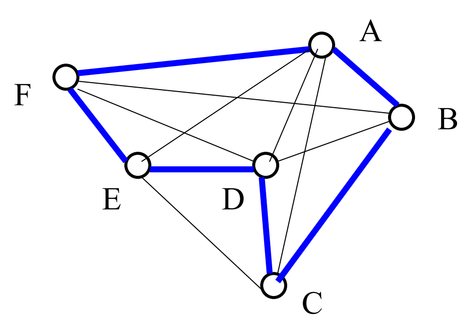
Quality measure

- reflects our **preference towards each configuration** (or state)

Example: Traveling salesman problem

Problem:

- A graph with distances

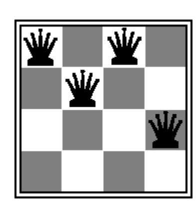


- **Goal:** find the shortest tour which visits every city once and returns to the start

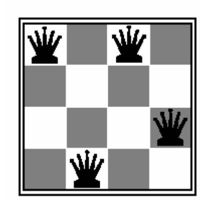
  An example of a valid **tour:** ABCDEF

Example: N queens

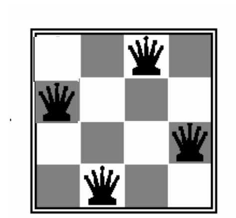
- A CSP problem can be converted to the 'optimal' configuration problem
- **The quality of a configuration in a CSP**
  = the number of constraints violated
- **Solving:** minimize the number of constraint violations



# of violations =3



# of violations =1



# of violations =0

Iterative optimization methods

- Solutions to **large 'optimal' configuration** problems are often found using **iterative optimization methods**
- **Why?**
  – Searching systematically for the best configuration with the **search methods** covered so far may not be the best solution
  – Running times of DFS and BFS:
    - Exponential in the number of variables
  – Uniform cost or A* algorithms would require
    - Too many partial solutions are kept active
- **Iterative Optimization Methods:**
  – **?**

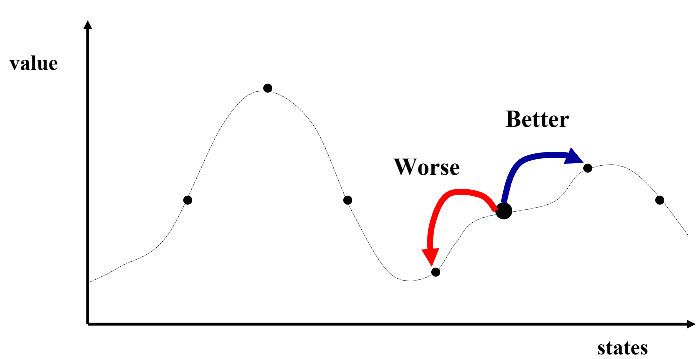
Iterative optimization methods

Properties:

- **Search the space of "complete" configurations**
- **Take advantage of local moves**
  - Operators make "local" changes to "complete" configurations
- **Keep track of just one state (the current state)**
  - no memory of past states
  - **!!! No search tree is necessary !!!**

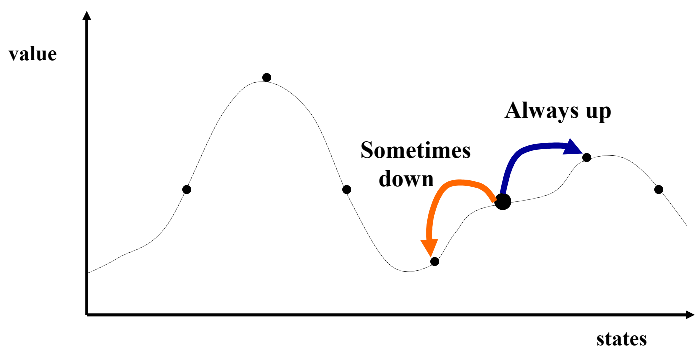
Hill climbing

- Look around at states in the local neighborhood and choose the one with the best value
- Problems: ?



Simulated annealing

- Permits “bad” moves to states with lower value, thus escape the local optima
- **Gradually decreases** the frequency of such moves and their size (parameter controlling it – **temperature**)



Simulated annealing algorithm

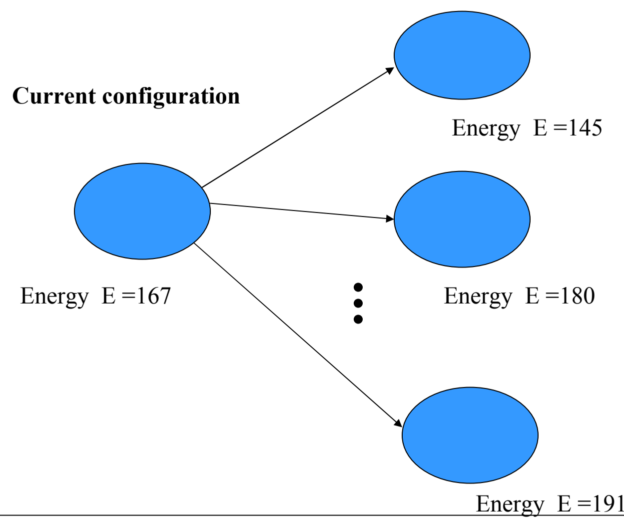
The probability of making a move:

- A good move (moving into a state with a higher value)
  - Probability is 1
- A “bad” move (moving into a state with a lower value)
  - is

$$p(Accept\ NEXT) = e^{\Delta E / T} \quad \text{where} \quad \Delta E = E_{NEXT} - E_{CURRENT}$$

- **Proportional to the energy difference**

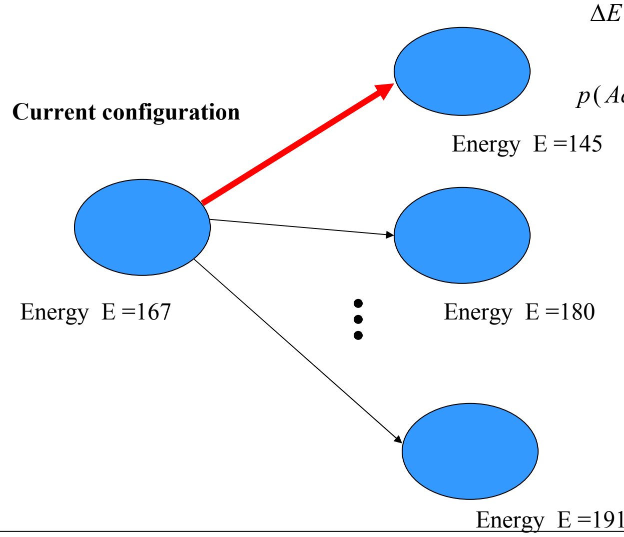
Simulated annealing algorithm



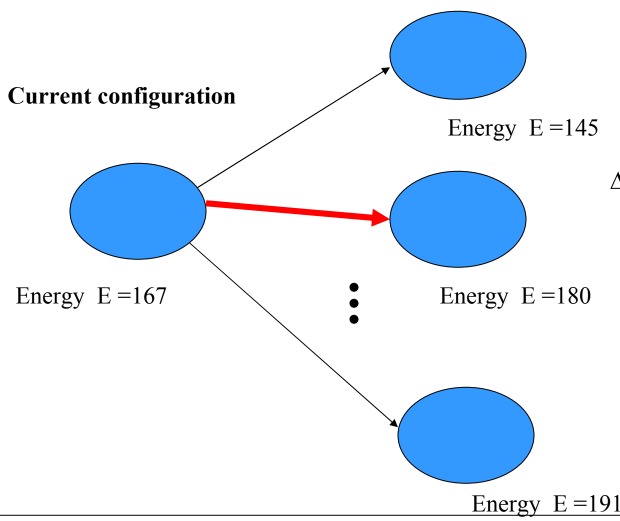
Simulated annealing algorithm

$$\Delta E = E_{NEXT} - E_{CURRENT}$$
$$= 145 - 167 = -22$$
$$p(Accept) = e^{\Delta E / T} = e^{-22/T}$$

Sometimes accept!



Simulated annealing algorithm



$$\Delta E = E_{NEXT} - E_{CURRENT}$$
$$= 180 - 167 > 0$$
$$p(Accept) = 1$$

Always accept!

Simulated annealing algorithm

The probability of moving into a state with a lower value is

$$p(Accept) = e^{\Delta E / T} \quad \text{where} \quad \Delta E = E_{NEXT} - E_{CURRENT}$$

The probability is:

- **Modulated through a temperature parameter T**:
  - for $T \to \infty$ the probability of any move approaches 1
  - for $T \to 0$ the probability that a state with smaller value is selected goes down and approaches 0
- **Cooling schedule:**
  - Schedule of changes of a parameter T over iteration steps

Simulated annealing algorithm

- **Simulated annealing algorithm**
  - developed originally for modeling physical processes (Metropolis et al, 53)
- **Properties:**
  - **If T is decreased slowly enough the best configuration (state) is always reached**
- **Applications:**
  - VLSI design
  - airline scheduling

Simulated evolution and genetic algorithms

- Limitations of **simulated annealing**:
  - Pursues one state configuration at the time;
  - Changes to configurations are typically local

Can we do better?

- Assume we have two configurations with good values that are quite different
- We expect that the combination of the two individual configurations may lead to a configuration with higher value (**Not guaranteed !!!**)

This is the idea behind **genetic algorithms** in which we grow a population of individual combinations

Genetic algorithms

Algorithm idea:

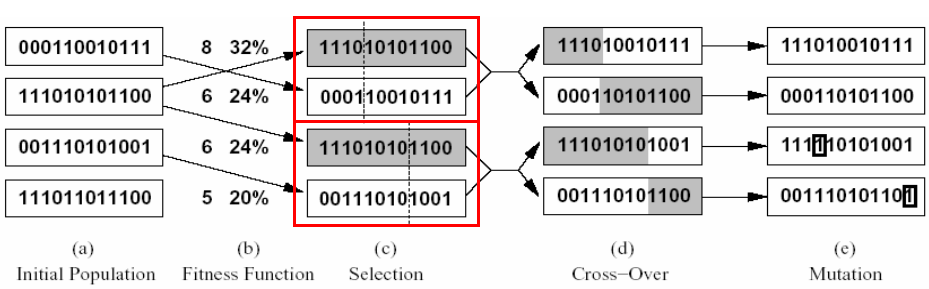
- **Create a population of random configurations**
- **Create a new population through:**
  - Biased selection of pairs of configurations from the previous population
  - Crossover (combination) of pairs
  - Mutation of resulting individuals
- **Evolve the population over multiple generation cycles**

- **Selection of configurations to be combined:**
  - **Fitness function = value function**

    measures the quality of an individual (a state) in the population

Reproduction process in GA

- Assume that a state configuration is defined by a set variables with two values, represented as 0 or 1



Parametric optimization

Optimal configuration search:

- Configurations are described in terms of variables and their values
- Each configuration has a quality measure
- Goal: find the configuration with the best value

When the state space we search is finite, the search problem is called a **combinatorial optimization problem**

When parameters we want to find are real-valued

- **parametric optimization problem**

Parametric optimization

Parametric optimization:

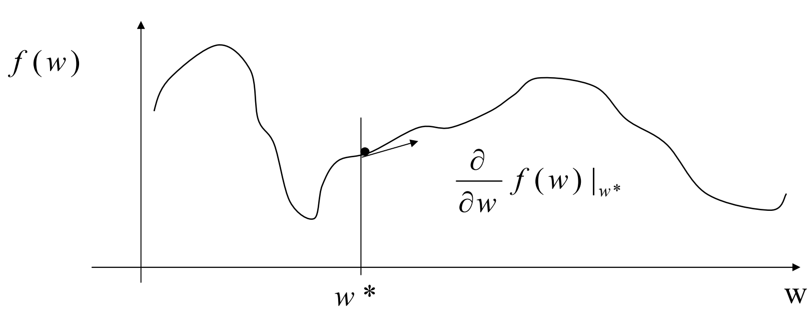
- Configurations are described by a vector of free parameters (variables) $\mathbf{w}$ with real-valued values
- **Goal:** find the set of parameters $\mathbf{w}$ that optimize the quality measure $f(\mathbf{w})$

Parametric optimization techniques

- **Special cases (with efficient solutions):**
  - **Linear programming**
  - **Quadratic programming**
- **First-order methods:**
  - **Gradient-ascent (descent)**
  - **Conjugate gradient**
- **Second-order methods:**
  - **Newton-Rhapson methods**
  - **Levenberg-Marquardt**
- **Constrained optimization:**
  - **Lagrange multipliers**

Gradient ascent method

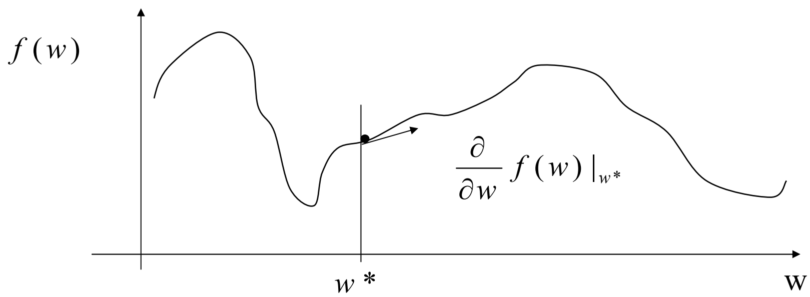
- **Gradient ascent:** the same as hill-climbing, but in the continuous parametric space **w**



- What is the derivative of an increasing function?

Gradient ascent method

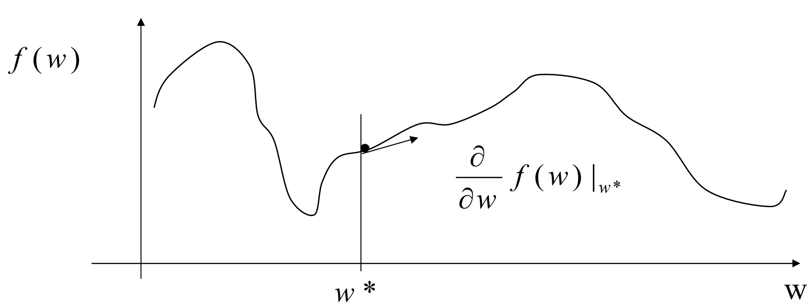
- **Gradient ascent:** the same as hill-climbing, but in the continuous parametric space **w**



- What is the derivative of an increasing function?
  - positive

Gradient ascent method

- **Gradient ascent:** the same as hill-climbing, but in the continuous parametric space **w**

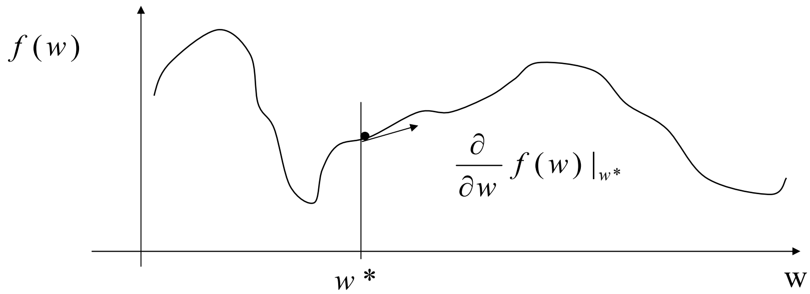


- Change the parameter value of w according to the gradient

$$w \leftarrow w^* + \alpha \frac{\partial}{\partial w} f(w)|_{w*}$$

CS 1571 Intro to AI M. Hauskrecht

Gradient ascent method



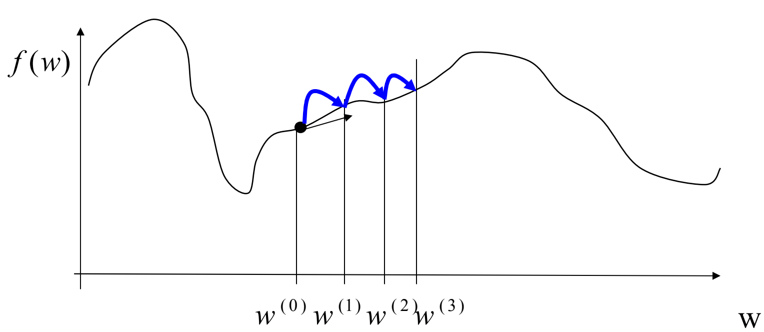
- New value of the parameter

$$w \leftarrow w^* + \alpha \frac{\partial}{\partial w} f(w)|_{w^*}$$

$\alpha > 0$ - a learning rate (scales the gradient changes)

Gradient ascent method

- To get to the function minimum repeat (iterate) the gradient based update few times



- **Problems:** local optima, saddle points, slow convergence
- More complex optimization techniques use additional information (e.g. second derivatives)