# Randomized search and web scalability

# Optimization problems: from Maths to CS to user experience

- optimization
  - knapsack
  - travel planning
  - Hiring (seen in lec. 4)
- clusterization
  - web search engines (clustering & ordering)
- classification
  - voice/face recognition
  - word best match: soundness, ortographic correction

# Operational research

It is desirable to model choice as a minimization problem.

- Given a set of possible solutions (search space) find the best solution that maximize/minimize a "energy"/"cost" function.
  - given a set of bags, find the best arrangement in the trunk of your car
  - given a set of apples of variouse size and a price table that define the price respect of the minimal size, find the best size that maximize earnings.

# Optimal Solution vs Approximation

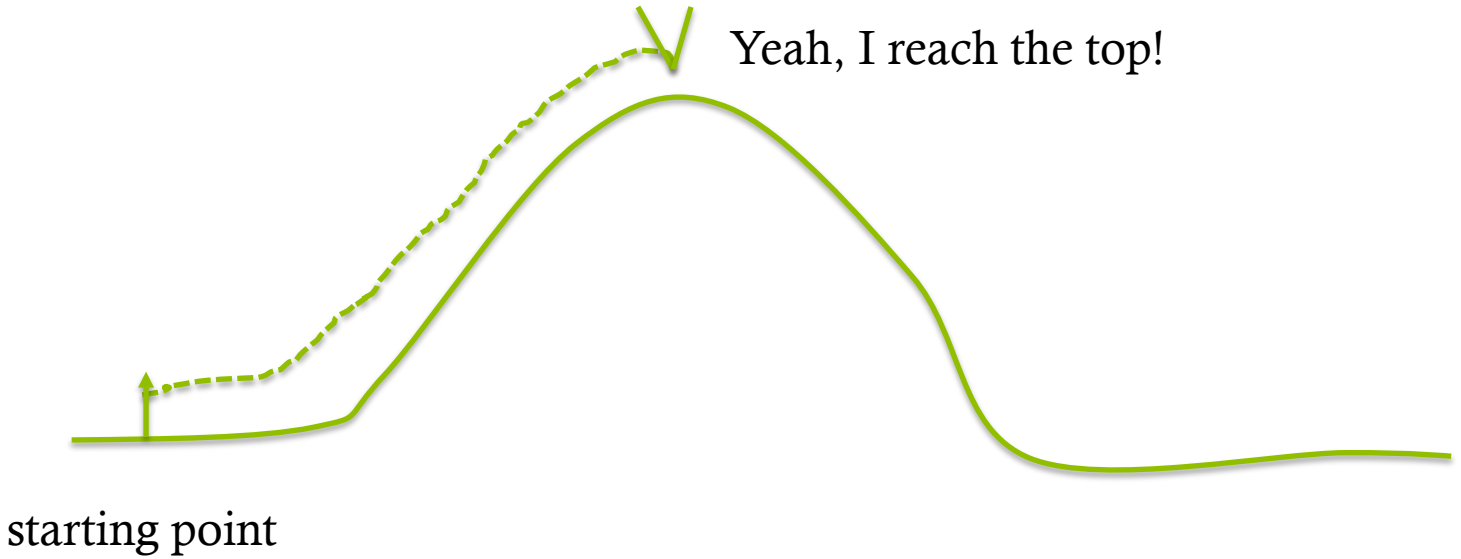
For almost all "interesting" problems, it is "hard" to find the best solution.

We have to find something similar (near) to the best.

We need a search path for a good approximation

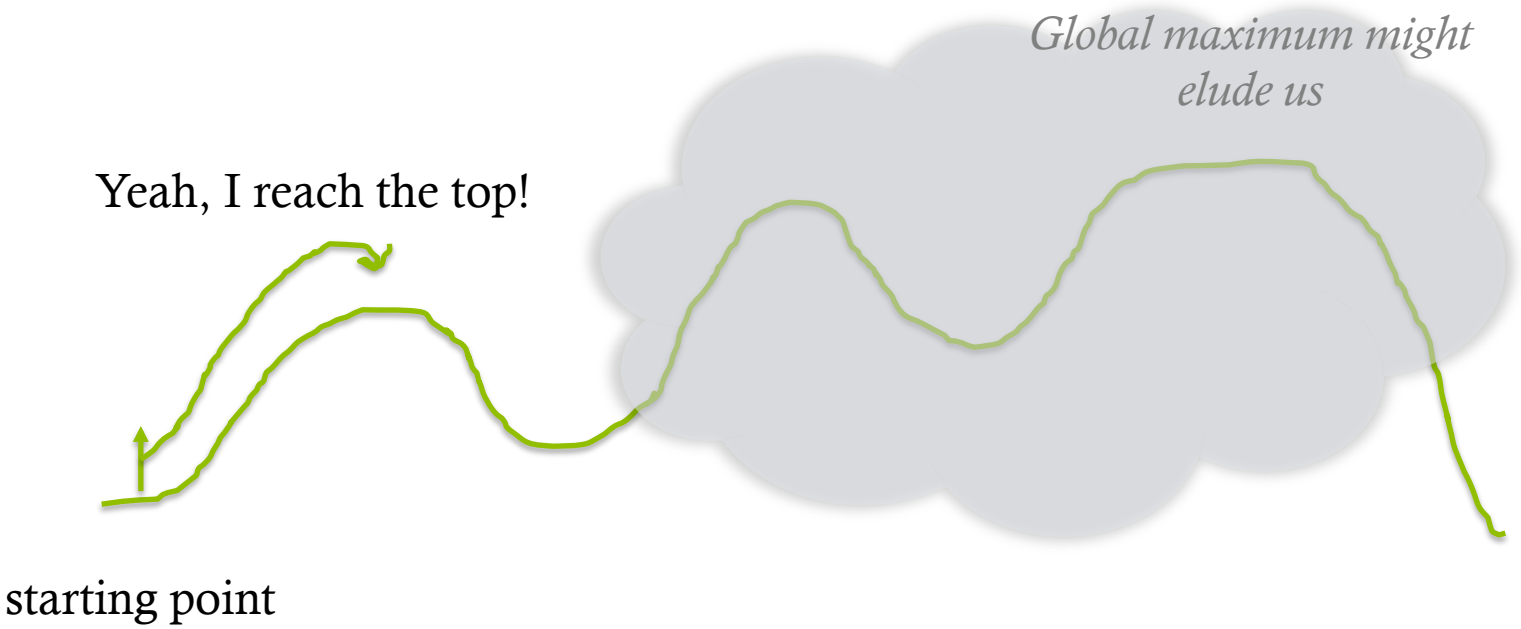
# Greedy Methods

- Utility function is way different from a straight line. Search by always improving on the local condition



# The local-max problem

- If only I could see farther…

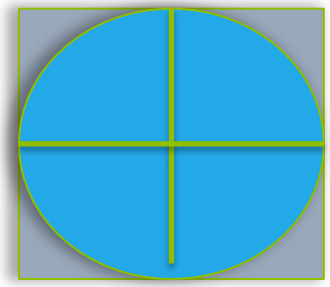


# Using chance/randomness to compute

Montecarlo methods: use frequencies to approximate probabilities to then compute exact values.

- [Buffon, 1733 circa] calculates π by a physical experiment:
- uniform randomly take a set of point inside a square containing a circle, if it is inside (easy to calculates) I count it on the area of the circle and the square, otherwise I count it only for the square.

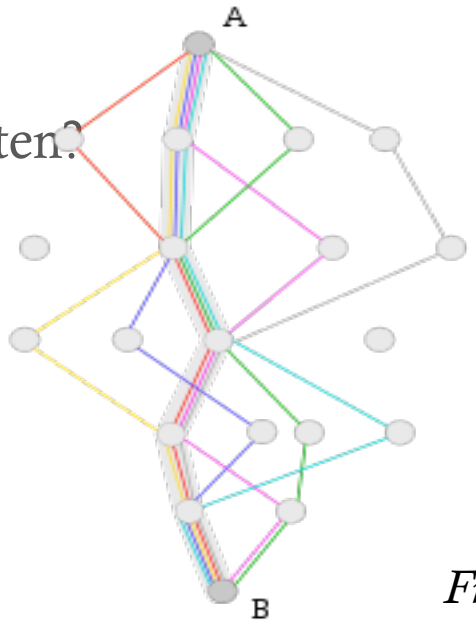
$A_{circle}=\pi r^2$



$A_{square}=(2r)^2=4r^2$

# Ant colonies (best path)

- more randomly around, sign the local best paths and (likely) continue search on it.
  - Shall the best path be visited more often?
  - Individuals are “weak” but
  1) highly collaborative
  2) Deliberately apply randomness




*From Wikipedia*

# Genetic algorithms

Probably the best (random) algorithm inspired by nature

- Encode an individual's features as a binary string: the DNA
- Formalize fitness to solve a task a function: DNA->number
- Take a set of good candidates, the parent generation
- Create offsprings by randomly combining parents' DNA + random mutations: the children generation
- Repeat generations in the hope some *uber-individual* will arise

# Genetic: How it Works!

Given a generation $G_n$, the next $G_{n+1}$ is obtained by union of:

- THE WINNERS: some actual "best fitting" individual (not all, not the bestest)
- THE SONS: some new individual obtained by join pieces of other individuals (crossover)
- THE MUTANTS: some actual individual in which we have made some radomn changes (mutations)

- ...and the first generation $G_0$? randomly chosen? crafted selections? Does not matter!

# Consequences

- On the Web, it's hard to assess what is the best solution and what value it should have
- Traditional applied Mathematics does not directly apply.
- On-line aspect: the Secretary problem
- Incomplete information: the greedy search
- The [web] scalability/big data aspect: apply nature-inspired methods and hope to get the best or near best solutions.