Optimization

- Exhaustive search
- Greedy search and hill climbing
- Gradient ascent
- Simulated annealing

Optimization

We need

- A numerical representation x for all possible solutions to the problem
- A function $f(x)$ that tells us how good solution x is
- A way of finding
  - $\max_x f(x)$ if bigger $f(x)$ is better (benefit)
  - $\min_x f(x)$ if smaller $f(x)$ is better (cost)

Discrete optimization

- **Chip design**
  - Routing tracks during chip layout design
- **Timetabling**
  - E.g.: Find a course time table with the minimum number of clashes for registered students
- **Travelling salesman problem**
  - Optimization of travel routes and similar logistics problems

Exhaustive search

- Test all possible solutions, pick the best
- Guaranteed to find the optimal solution

Exhaustive search

Only works for simple discrete problems, but can be approximated in continuous problems

- Sample the space at regular intervals (grid search)
- Sample the space randomly N times

UiO : Department of Informatics
University of Oslo

Greedy search

- Pick a solution as the current best
- Compare to all neighboring solutions
  - If no neighbor is better, then terminate
  - Otherwise, replace the current best with the best of the neighbors
  - Repeat

UiO : Department of Informatics
University of Oslo

Hill climbing

- Pick a solution as the current best
- Compare to a random neighbor
  - If the neighbor is better, replace the current best
  - Repeat

UiO : Department of Informatics
University of Oslo

Continuous optimization

- **Mechanics**
  - Optimized design of mechanical shapes etc.
- **Economics**
  - Portfolio selection, pricing options, risk management etc.
- **Control engineering**
  - Process engineering, robotics etc.

Gradient ascent / descent

In continuous optimization we may be able to calculate the gradient of $f(x)$:

$$\nabla f(x) = \begin{bmatrix} \frac{\delta f(x)}{\delta x_0} \\ \frac{\delta f(x)}{\delta x_1} \\ \vdots \\ \frac{\delta f(x)}{\delta x_n} \end{bmatrix}$$

The gradient tells us in which direction $f(x)$ increases the most

UiO : Department of Informatics
University of Oslo

Gradient ascent / descent

Starting from $x^{(0)}$, we can iteratively find higher $f\left(x^{(k+1)}\right)$ by adding a value proportional to the gradient to $x^{(k)}$:

$$x^{(k+1)} = x^{(k)} + \gamma \nabla f\left(x^{(k)}\right)$$

UiO : Department of Informatics
University of Oslo

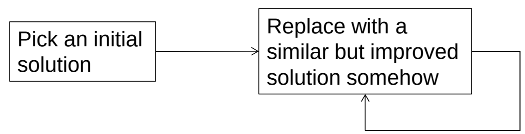
Local optima

Algorithms like greedy search, hill climbing and gradient can only find local optima:

- They will only move through a strictly improving chain of neighbors
- Once they find a solution with no better neighbors they stop

UiO : Department of Informatics
University of Oslo

Exploitation



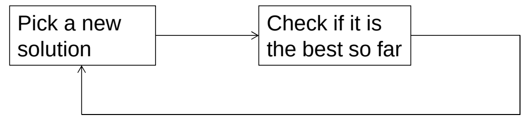
UiO : Department of Informatics
University of Oslo

Global optimization

- Most of the time, we must expect the problem to have many local optima
- Ideally, we want to find the best local optima: the global optimum

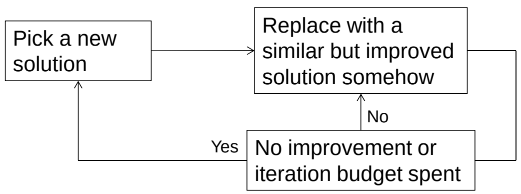
UiO : Department of Informatics
University of Oslo

Exhaustive search – pure exploration



UiO : Department of Informatics
University of Oslo

Mixed solution



Works better, but only if there are few local optima

UiO : Department of Informatics
University of Oslo

Going the wrong way

What if we modified the hill climber to sometimes choose worse solutions?

- Goal: avoid getting stuck in a local optimum
- Always keep the new solution if it is better
- However, if it is worse, we'd still want to keep it sometimes, i.e. with some probability

Annealing

A thermal process for obtaining low energy states of a solid in a heat bath:

- Increase the temperature of the heat bath to a the point at which the solid melts
- Decrease the temperature slowly
- If done slowly enough, the particles arrange themselves in the minimum energy state

UiO : Department of Informatics
University of Oslo

Simulated annealing

- Set an initial temperature T
- Pick an initial solution
- Repeat:
  - Pick a solution neighboring the current solution
  - If the new one is better, keep it
  - Otherwise, keep the new one with a probability
    $$P(\Delta f, T) = e^{-\Delta f/T}$$
  - Decrease T

UiO : Department of Informatics
University of Oslo

Simulated Annealing Illustrated