Optimisierung mit Particle Swarms
Wie geht das?

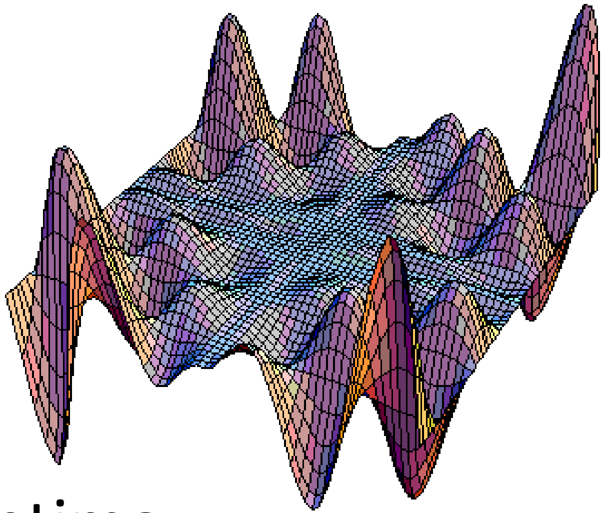
Brown Bag 1.9.2020 – Dr. Sven Magg

Problem Definition

We have an **optimisation problem**, that…

- Lets us compute a value at each position (the value we want to optimise)
- Has a continuous search space (usually vectors in cartesian space)

- Can be high-dimensional
- Also variants for discrete, multi-objective, adaptive, etc problems



e.g.: (Hyper-)Parameter Optimisation, search for global optima, …

Basic Idea

We have a flock of birds collectively searching for food

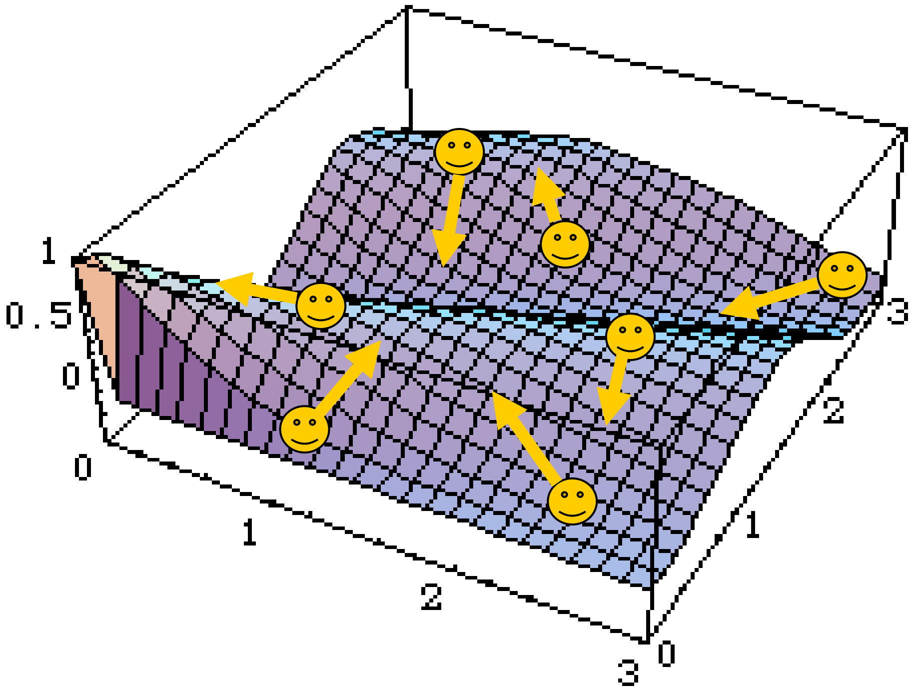
The flock is most likely to succeed when birds combine **three strategies**:

1. **Brave**:
   keep flying in the same direction
2. **Conservative**:
   fly back towards its own best previous position
3. **Swarm**:
   move towards its best neighbour

HITeC

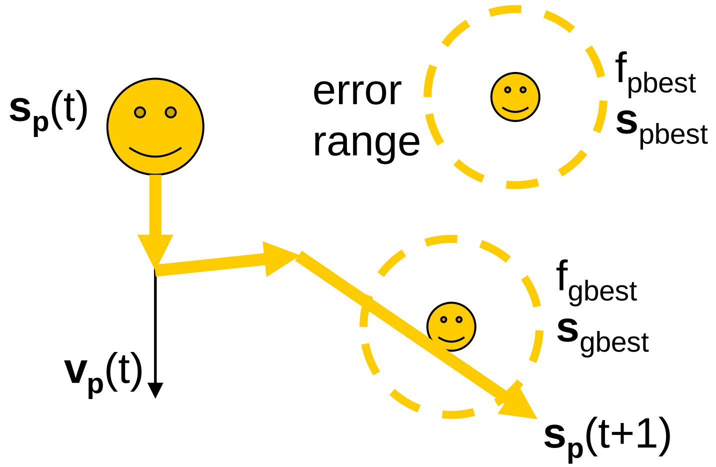
Particle Swarm Optimisation (PSO)

- Proposed by Kennedy and Eberhart (2001)
- Optimisation search space with dimension n
- Flock now vector of particles p with
  - **Position s**
  - **Velocity v**
  - **Performance f**
- All particles
  - perceive f and s of neighbouring particles
  - can select best neighbour (gbest)
  - remember own best position so far (pbest)



Particle Update

- A particle computes next position by taking into account
  - Fraction of own current velocity v
  - Direction to own previous best
  - Direction to best neighbour
  - (Some error for *gbest* and *pbest)*



$$\mathbf{v}_p(t+1) = a \times \mathbf{v}_p(t) + b \times R \times (\mathbf{s}_{pbest} - \mathbf{s}_p(t)) + c \times R \times (\mathbf{s}_{gbest} - \mathbf{s}_p(t))$$

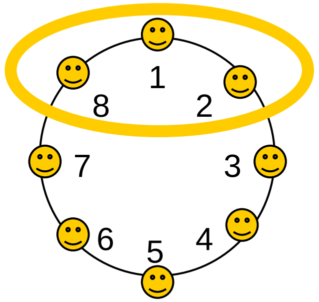
where a, b, c are constants between 0 and whatever
(often also ω, C_1, C_2 , a.k.a intertia weight, cognitive factor, social factor)
R is a random number between 0 and 1

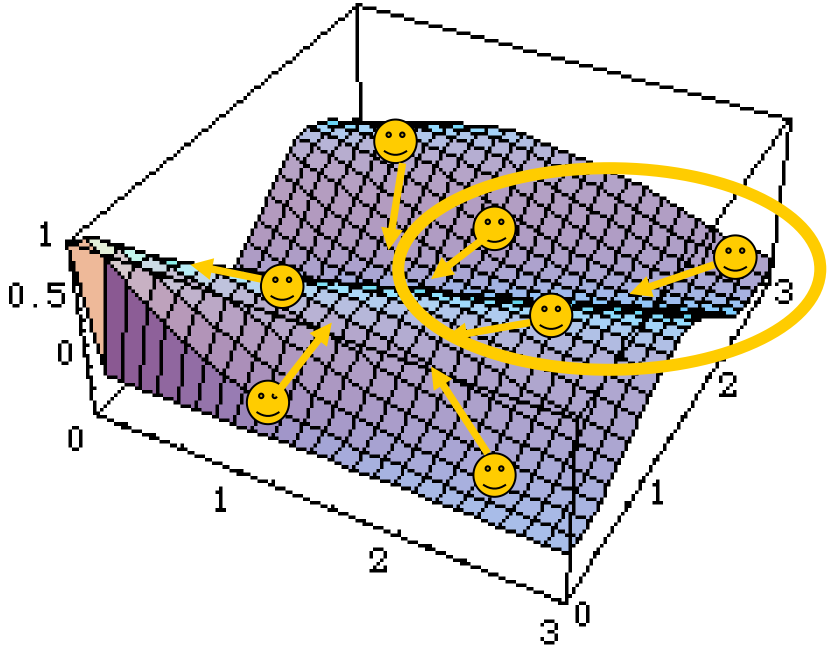
$$\mathbf{s}_p(t+1) = \mathbf{s}_p(t) + \mathbf{v}_p(t+1)$$

Neighbourhoods

Neighbourhoods can be defined:

- Complete (All particles are neighbours)
- Geographical (local) neighbourhood
- Social neighbourhood





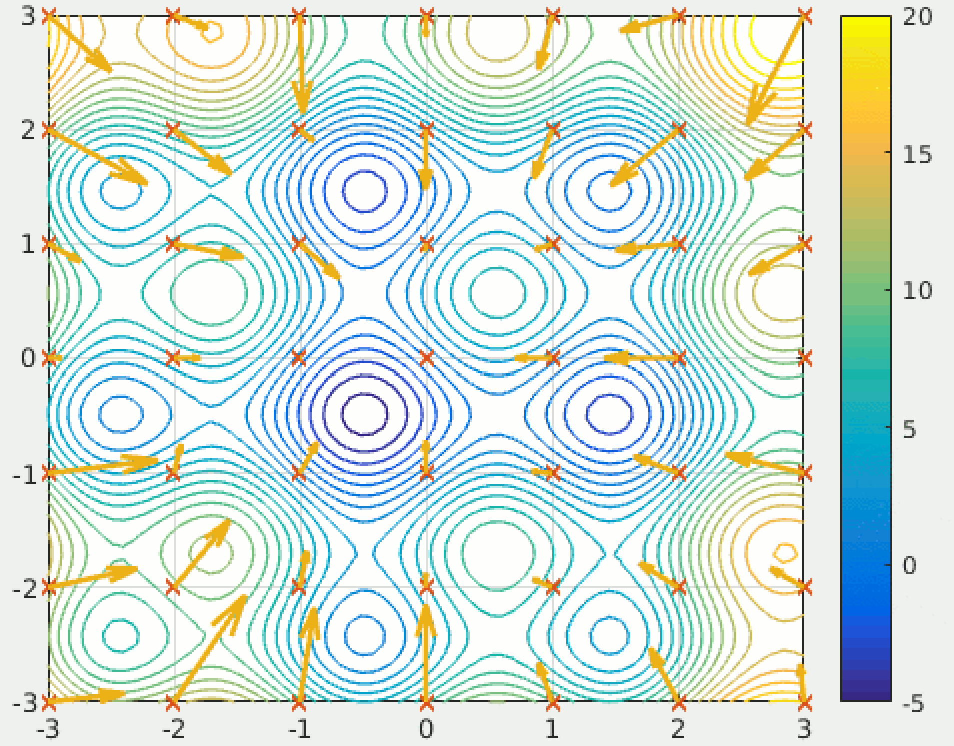
Algorithm

1. Initialise
    - Initial random s and v, set *pbest* to initial position
    - Typically 20 particles for problems with dimensionality 2 – 200
    - Neighbourhood size, typically 3 to 5
2. Update particle velocities
3. Update particle positions
4. Iterate until solution *max(pbest)* acceptable or no further improvement

Search Dynamics

Depending on parameters, the following behaviour is observable:

- Particles quickly converge to each other
  - i.e. usually move towards centre of search space
- Insects around a light bulb
  - Don‘t converge to optimum directly but circle around it
  - Depends on inertia factor



From Wikimedia Commons

PSO vs. Evolutionionary Optimisation

PSO

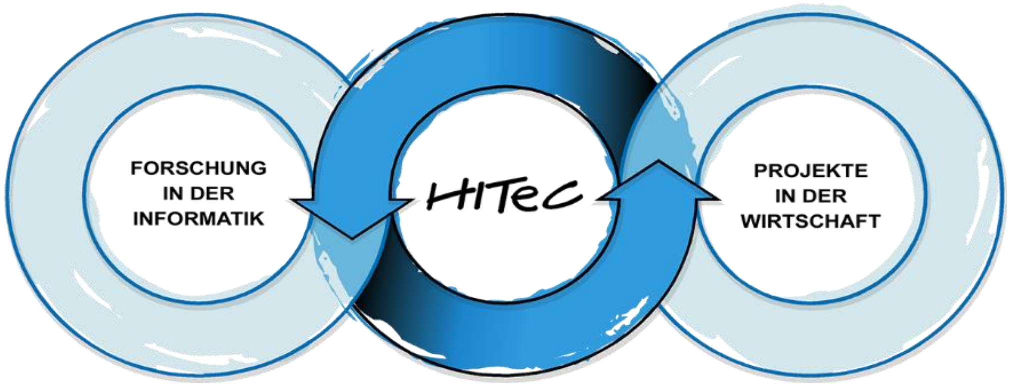
- Needs good gradients
- Continuous space
- Particles have direction and move collectively through the space (i.e. are always affected by others)

- Explores first the space between the particles

EVO

- Works also without good gradients
- Works on non-continuous spaces
- Individuals are fixed to a location, and children are created nearby (mostly), i.e. are independent

- Explores first surroundings of lucky individuals



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Eine Initiative des Fachbereichs
Informatik der Universität Hamburg

Vielen Dank für Eure Aufmerksamkeit!

Kontakt: Dr. Sven Magg, magg@informatik.uni-hamburg.de

Webseiten: hitec-hamburg.de
informatik.uni-hamburg.de
aric-hamburg.de