# Particle Swarms

Rui Mendes

Algoritmi
Departamento de Informática
Universidade do Minho

April 28, 2018

# Part I

# Particle Swarms

# Overview of Particle Swarms

- Created in 1995 by Jim Kennedy and Russ Eberhart
- Initially intended as a simulation of bird flocks
- Optimization algorithm
- Inspired in social dynamics
- Individuals preserve the memory of their best achievement
- Individuals imitate their most successful neighbor

# The particle swarm framework

1. Initialize the population
2. Evaluate the fitness of the population
3. Choose individuals from the neighborhood
4. Imitate those individuals
5. Update the best experience if the current one is better
6. Loop to 2

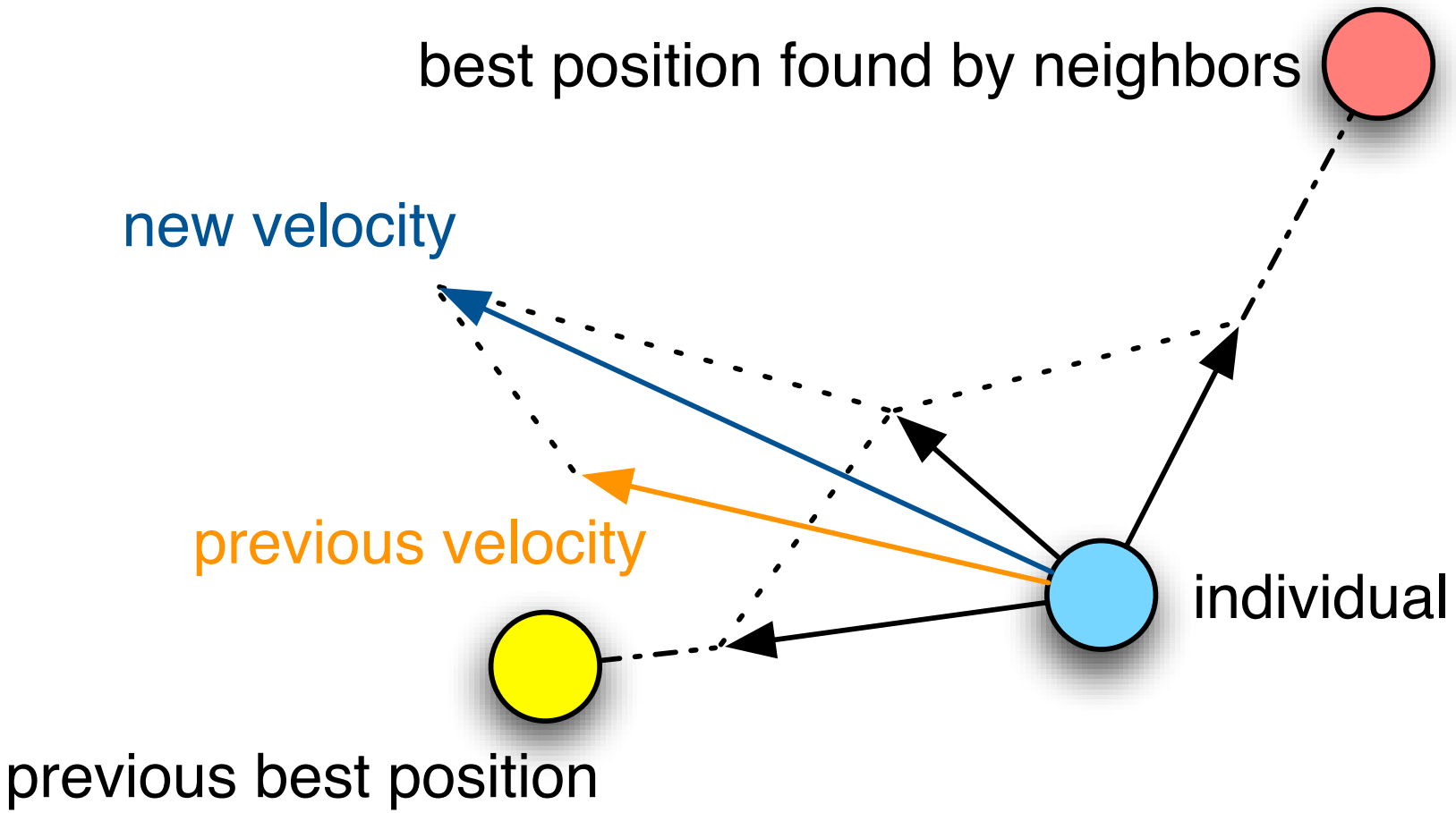
# The initial PSO algorithm

| | |
|---|---|
| Position | The current position $\vec{x_i}$ |
| Velocity | The current velocity $\vec{v_i}$ |
| Individualism | The previous best position $\vec{p_i}$ |
| Conformism | The best position found by their neighbors $\vec{p_g}$ |
| $\varphi_1$ and $\varphi_2$ | Weighting of the stochastic acceleration terms |

## Particle Swarm

$$\begin{cases} \vec{v_i} = & \vec{v_i} + \vec{\mathrm{U}}[0, \varphi_1](\vec{p_i} - \vec{x_i}) + \vec{\mathrm{U}}[0, \varphi_2](\vec{p_g} - \vec{x_i}) \\ \vec{x_i} = & \vec{x_i} + \vec{v_i} \end{cases}$$

# Solution generation in PSO



# Maximum velocity

- The velocity often becomes very large
- To counter this effect, a new parameter was introduced: $V_{max}$
- This parameter prevents the velocity from becoming too large
- This parameter is usually coordinate-wise
- If it is too large, individuals fly past good solutions
- If it is too small, individuals explore too slowly and may become trapped in local optima
- Early experience showed that $\varphi_1$ and $\varphi_2$ could be set to 2 for almost all applications and only $V_{max}$ needed to be adjusted

# Inertia weight

- Researchers soon became dissatisfied by the arbitrarity of $V_{max}$
- The individual's trajectories failed to converge
- $V_{max}$ disabled the particle's oscillations from exploration to exploitation

## Particle Swarm with inertia weight

$$\begin{cases} \vec{v_i} = & \alpha\vec{v_i} + \vec{\mathrm{U}}[0,\varphi_1](\vec{p_i} - \vec{x_i}) + \vec{\mathrm{U}}[0,\varphi_2](\vec{p_g} - \vec{x_i}) \\ \vec{x_i} = & \vec{x_i} + \vec{v_i} \end{cases}$$

- Proposed by Shi and Eberhart
- Initially linearly decreasing from 0.9 to 0.4
- $V_{max}$ is usually set to the range of the coordinate
- Fuzzy controllers were used to dynamically adapt it
- Later used as a random value around 0.5

# Constriction coefficient

- Clerc performed an analysis by studying the deterministic system
- He produced a generalized particle swarm model and studied its convergence properties
- From this he proposed a simplified model

## Particle Swarm with constriction coefficient

$$\begin{cases} \vec{v}_i = & \chi(\vec{v}_i + \vec{\mathrm{U}}[0, \varphi_1](\vec{p}_i - \vec{x}_i) + \vec{\mathrm{U}}[0, \varphi_2](\vec{p}_g - \vec{x}_i)) \\ \vec{x}_i = & \vec{x}_i + \vec{v}_i \end{cases}$$

where

$$\chi = \frac{2k}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$$

and $\varphi = \varphi_1 + \varphi_2$, $\varphi > 4$, $k \in [0, 1]$

# Equivalence between inertia and constriction models

- The inertia and constriction models are equivalent
- Clerc's results may also be applied to the inertia model
- Kennedy suggested using the constriction model and setting $\varphi$ to 4.1 ($\varphi_1 = \varphi_2 = 2.05$) and $k = 1$
- This determines $\chi \approx 0.729$
- This is equivalent to using the inertia model with $\alpha \approx 0.729$ and $\varphi_1 = \varphi_2 \approx 1.49445$
- There is no longer need to set $V_{max}$

# Canonical PSO

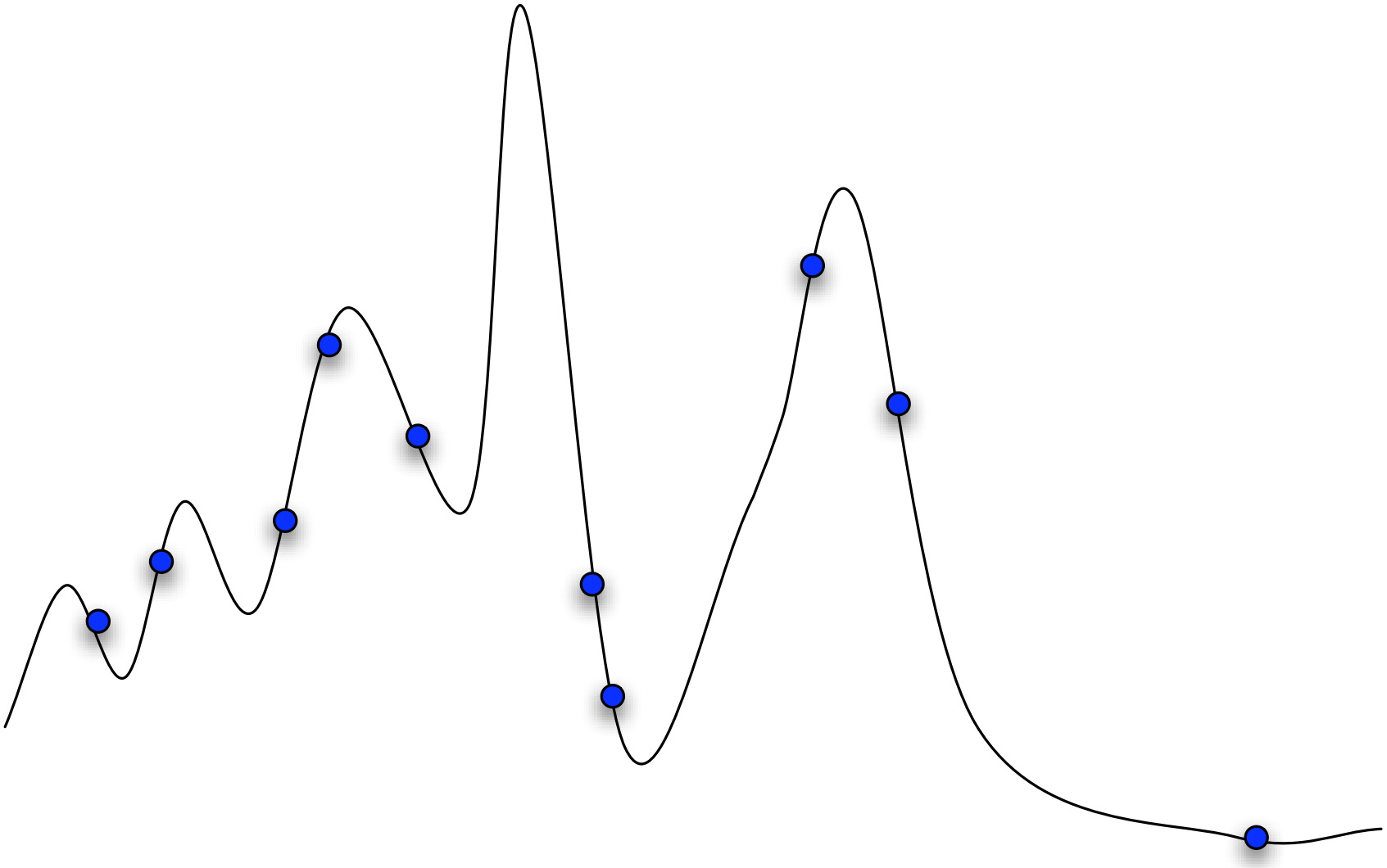
| | |
|---|---|
| Position | The current position $\vec{x}_i$ |
| Velocity | The current velocity $\vec{v}_i$ |
| Individualism | The previous best position $\vec{p}_i$ |
| Conformism | The best position found by their neighbors $\vec{p}_g$ |

## Canonical Particle Swarm

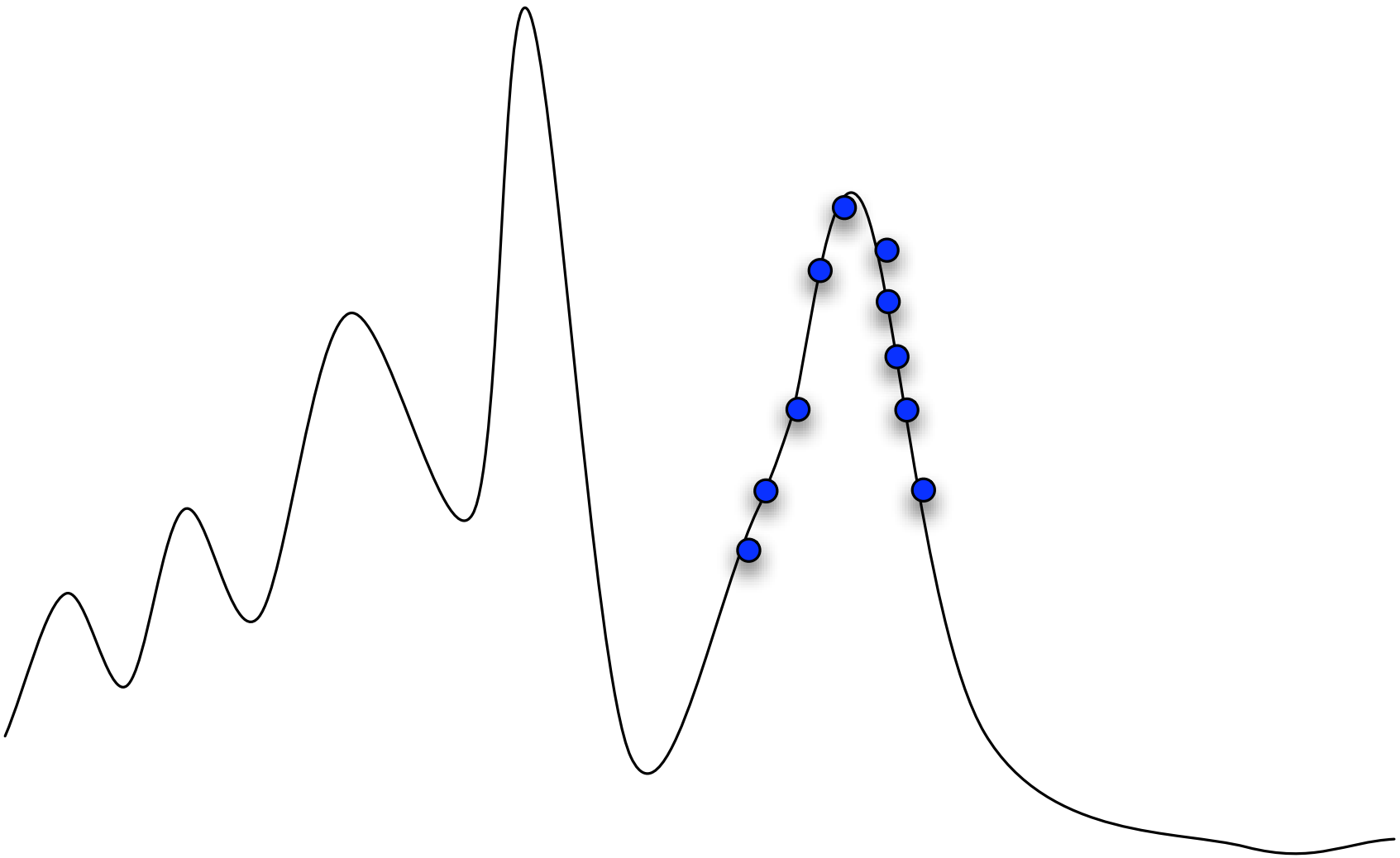
$$\begin{cases} \vec{v}_i = & \chi(\vec{v}_i + \vec{\mathrm{U}}[0, \varphi_1](\vec{p}_i - \vec{x}_i) + \vec{\mathrm{U}}[0, \varphi_2](\vec{p}_g - \vec{x}_i)) \\ \vec{x}_i = & \vec{x}_i + \vec{v}_i \end{cases}$$

where $\varphi_1 = \varphi_2 = 2.05, \chi = 0.729$

# What is premature convergence?



# What is premature convergence?



# What are the causes of premature convergence?

Optimization is a balance between two factors:

**exploration** The ability to explore the search space to find promising areas

**exploitation** The ability to concentrate on the promising areas of the search space

- An algorithm with too much exploration isn't *efficient*
- An algorithm with too much exploitation loses *diversity* fast
- If an algorithm doesn't have enough diversity, it will quickly stagnate

# How to solve premature convergence problems in PSO?

- Particle swarms have a tendency to converge prematurely
- There are two ways to help solve premature convergence problems
- To increase diversity in the population
- To understand the causes and remove them

# Self-organized criticality

- Proposed by Løvjberg and Krink
- When a particle is too close to another, its critical value increases
- The critical value is reduced each iteration by a certain amount to prevent it from building up unnecessarily
- When a particle's critical value reaches a certain threshold, it discharges, dispersing its criticality to the nearby particles and relocates itself

# Collision avoiding swarms

- Proposed by Blackwell and Bentley
- Used to track moving targets
- Proposed by Krink, Vesterstrøm and Riget
- Called spatially extended particles
- Particles bounce off when they collide

# Function stretching

- Proposed by Parsopoulos et al
- Perform a two-stage transformation of the objective function after detecting stagnation in a local minimum
- Elevate the function to eliminate all local minima below the one just found
- Stretch the area around the local minimum down
- Both stages do not alter the function for the regions above the local minimum
- After the transformation, the particles' position and velocity are reinitialized
- This technique may introduce false local optima and misleading gradients

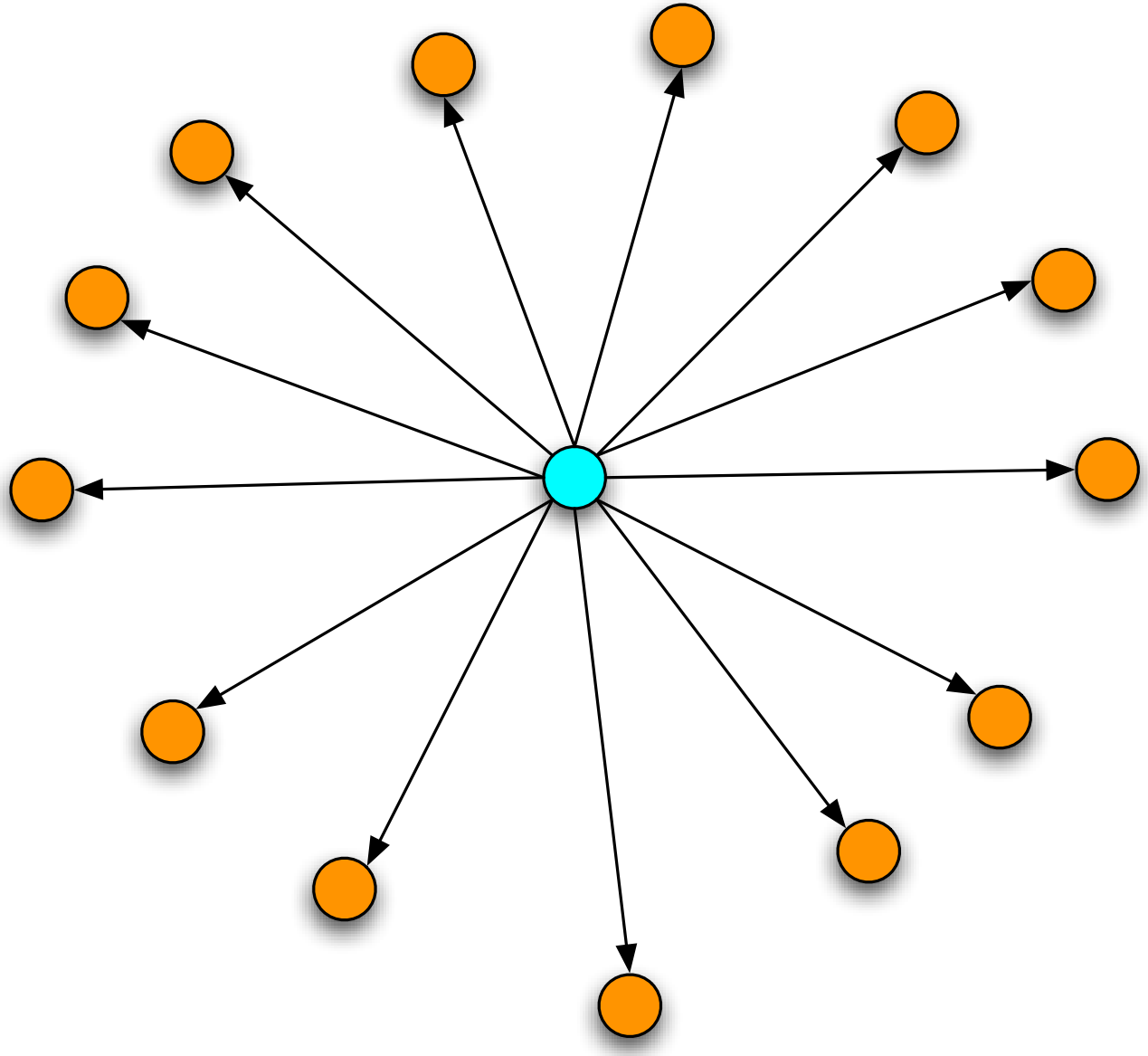
# Why is there premature convergence in PSO?

- Initially, the most successful individual in the population is responsible for social influence
- The good solution discovered attracts the whole population
- This may lead to premature convergence

# Why use neighborhoods

- Initially, there was no limited neighborhood
- The population used the *panmictic* model
- Everyone knows the entire population
- This model is known in PSO as global best (*gbest*)
- Convergence is fast
- Diversity is lost fast
- This leads to premature convergence

# Example of immediate influence



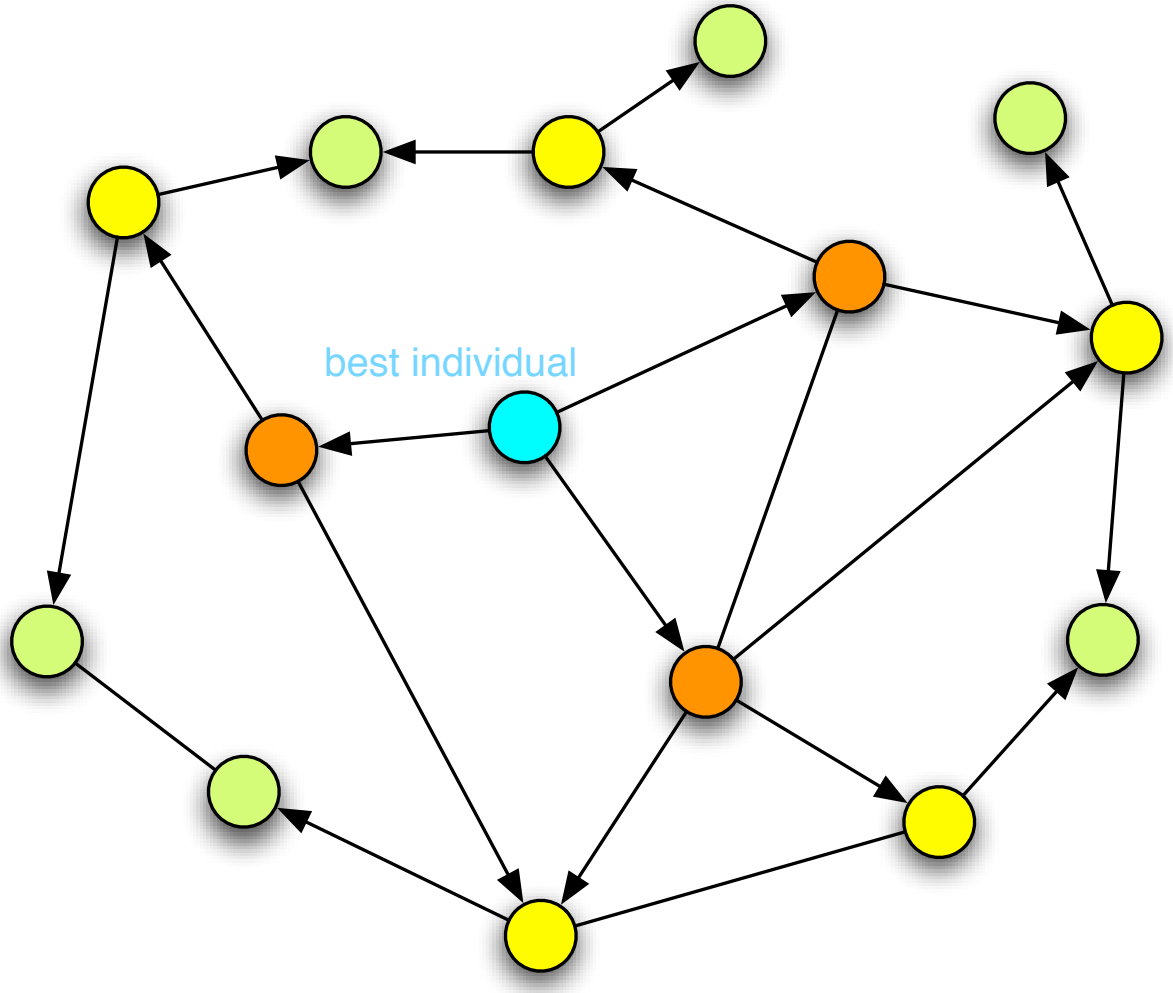
# Neighborhood concept

- Individuals imitate their (most successful) neighbors
- Each individual only directly influences his neighbors
- His neighbors will only influence their neighbors once they become sufficiently successful
- This favours clustering: different social neighborhoods may explore different areas of the search space
- Immediacy may be based on

  **Proximity** Proximity in Cartesian space

  **Social** To share social bonds

# Example of flow of influence




immediate
second degree
third degree

# Cartesian neighborhood

- Needs to compute distances between every pair of individuals
- Time complexity increases with population size and number of dimensions

# Social neighborhood

- The structure of the population is a social network
- Represented as a graph
  - Individuals are represented as vertices
  - Edges connect neighbors
- More efficient than cartesian neighborhood
- Richer structure

# Important graph statistics

**Degree** Average number of neighbors

**Distance** Average distance between two individuals

**Distribution sequence** Degree, number of second neighbors, number of third neighbors, etc

# Characteristics of a good topology

| | |
|---|---|
| Degree | relatively small, correlates highly with distance |
| Distance | neither too small nor too large |
| Distribution sequence | Largest proportion of the population should be in the first three values (cf. six degrees of sepparation) |

# Example of reasonable topologies

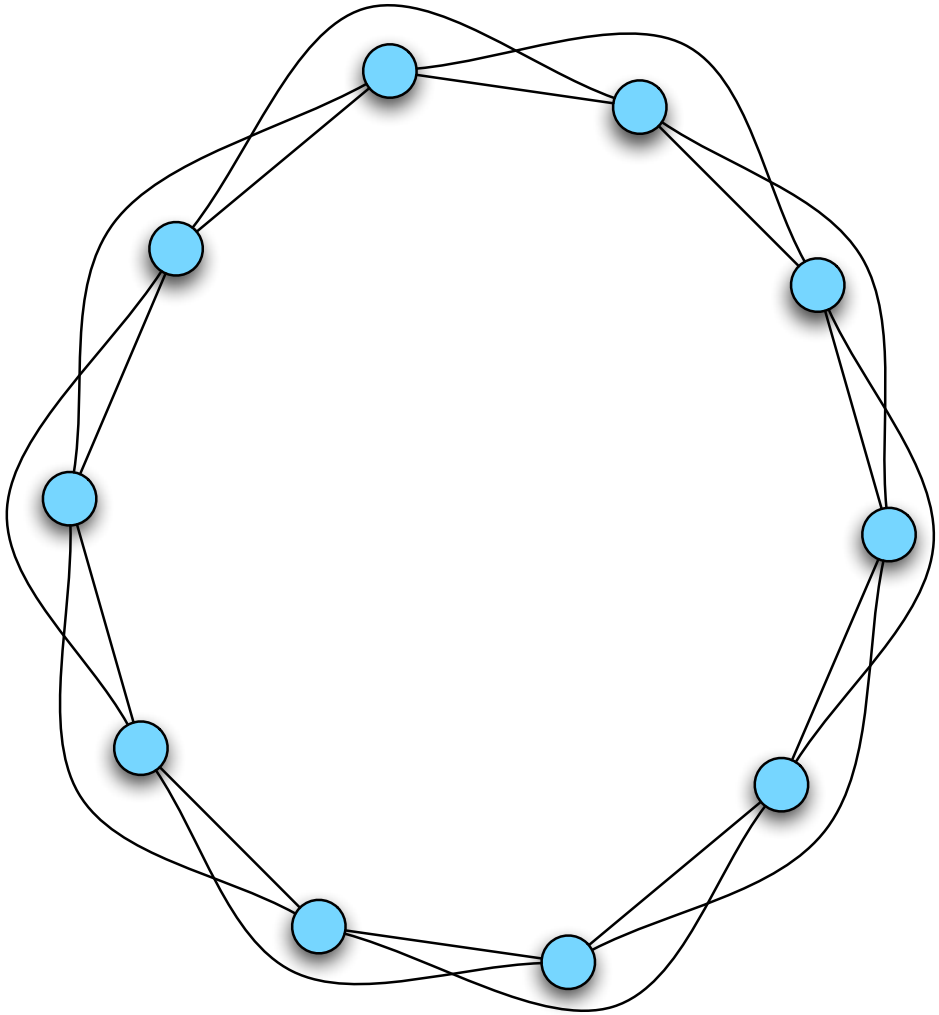


Figure: lbest2

# Example of reasonable topologies

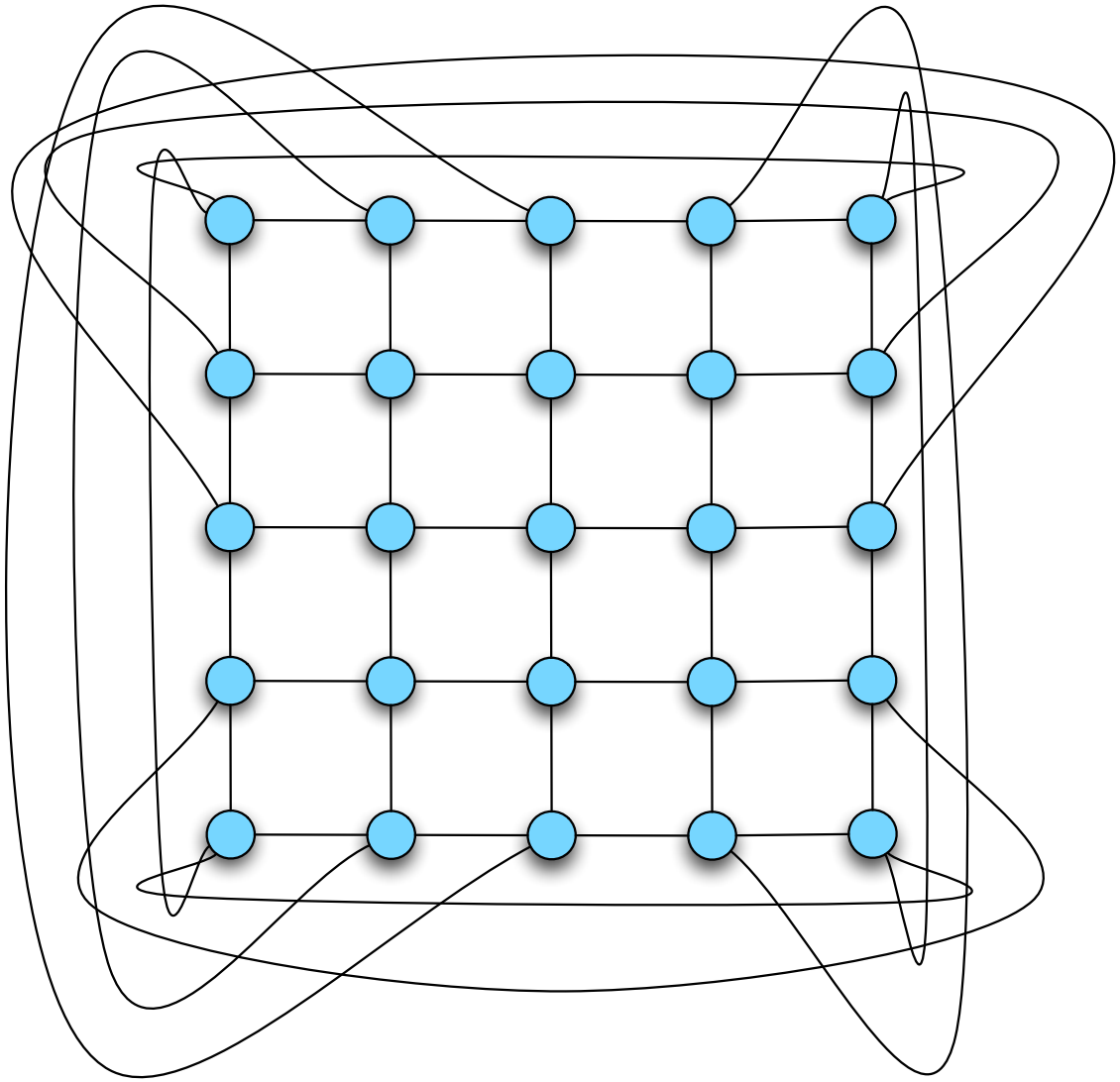


Figure: von Neumann

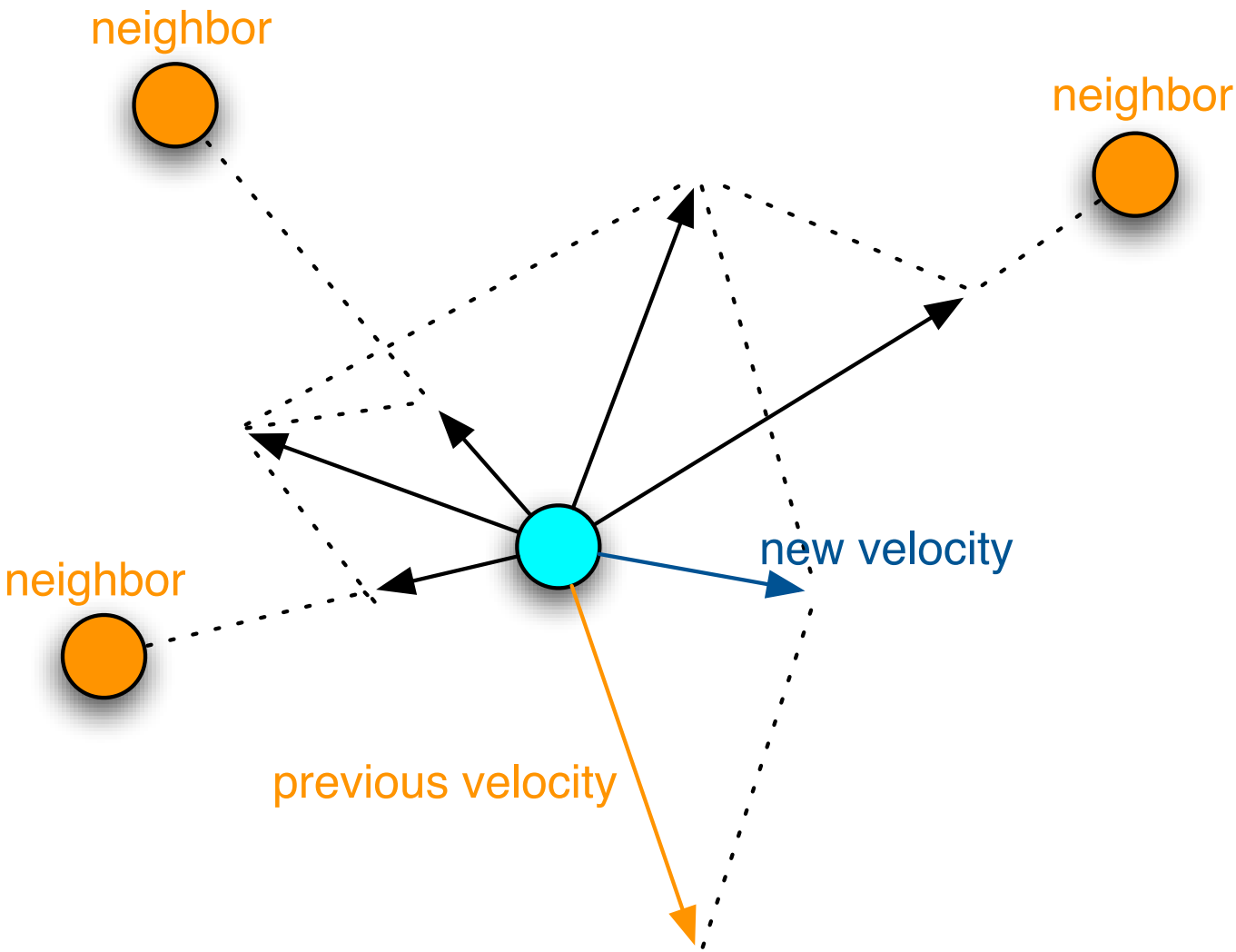
# Candidate Solution Generation for FIPS

1. All contributions of the neighborhood are used
2. Individual imitates the social norm
3. The social norm is the center of gravity
4. $\vec{p}_k$ is the best position of neighbor $k$
5. $\mathcal{N}$ is the set of neighbors

## FIPS

$$\begin{cases} \vec{v}_{t+1} = & \chi\left(\vec{v}_t + \frac{\sum_{k\in\mathcal{N}} \vec{\mathrm{U}}[0,\varphi_{max}](\vec{p}_k - \vec{x}_t)}{|\mathcal{N}|}\right) \\ \vec{x}_{t+1} = & \vec{x}_t + \vec{v}_{t+1} \end{cases}$$

# Solution generation in FIPS



# Differences in FIPS

- There is no self contribution
- All individuals in the neighborhood contribute to the influence
- The number of individuals used is very important: A few contributions, typically between 2 and 4 are best
- Given a well chosen population topology, it outperforms the canonical model

# Part II

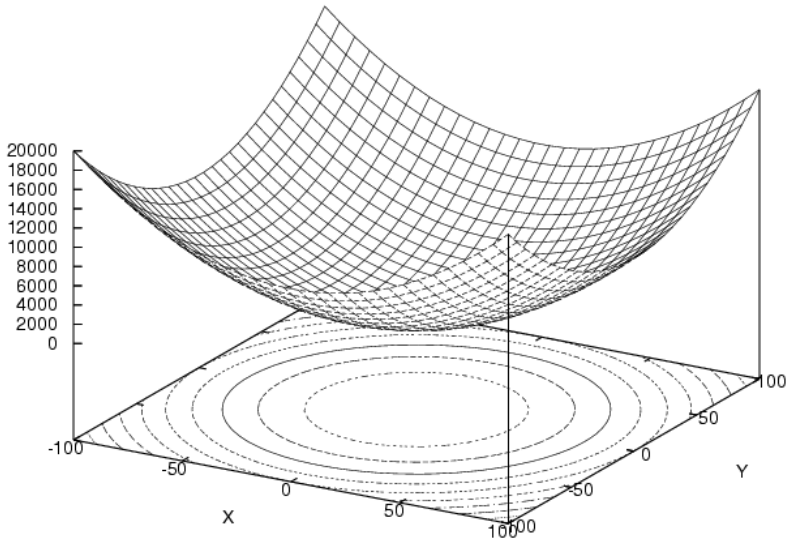
# Experiments

# The Function Set

- Commonly used to test optimization algorithms
- Multi-modality
- Deceptive gradient information
- The curse of dimensionality

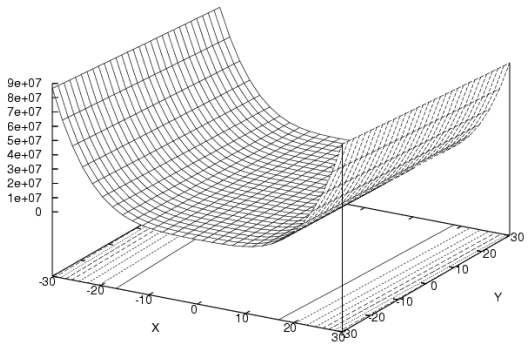
# The Sphere Function $f_1$

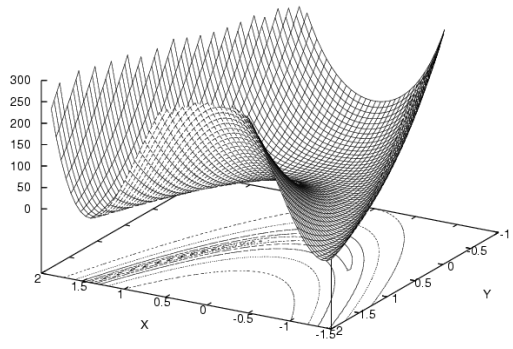
- Simple
- Unimodal



# The Rosenbrock Function $f_3$

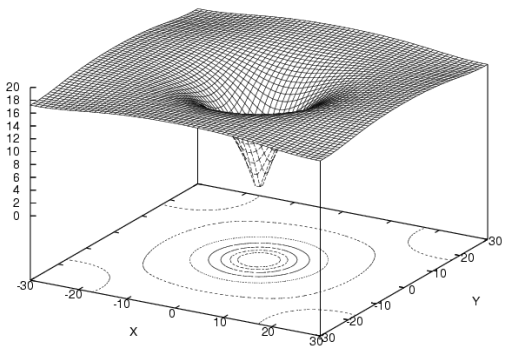
- Unimodal
- Strongly dependent variables
- Misleading gradient information

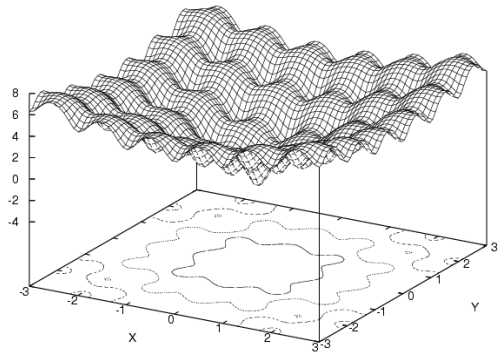


# The Ackley Function $f_6$

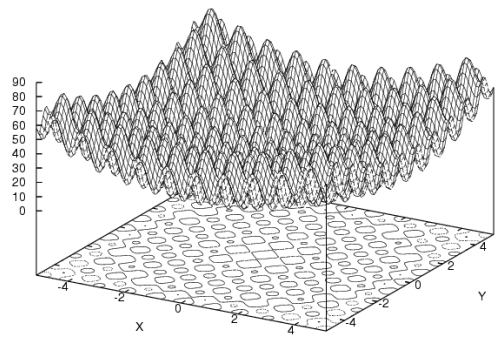
- Multimodal
- Many local optima

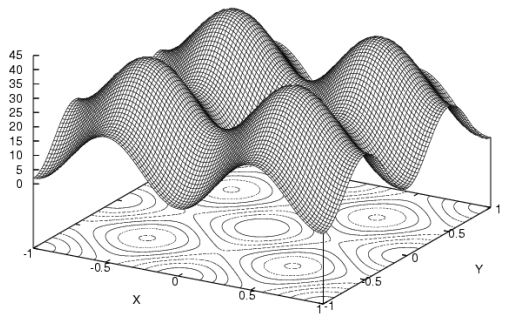


# The Rastrigin Function $f_4$

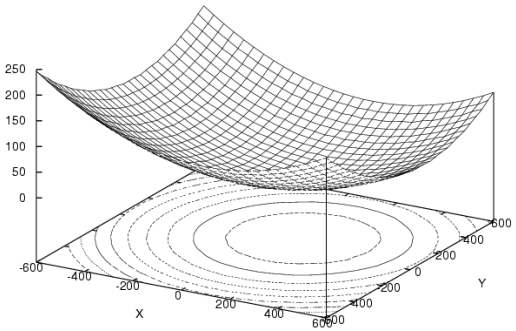
- Multimodal version of the Sphere
- Deep local minima arranged as sinusoidal bumps

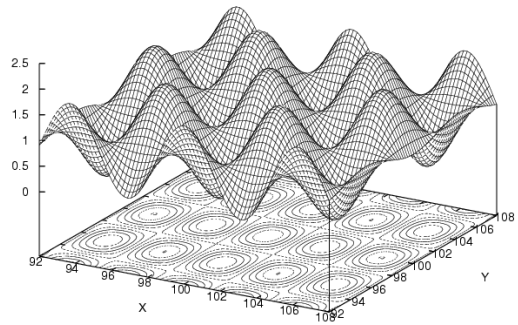


# The Griewank Function $f_5$

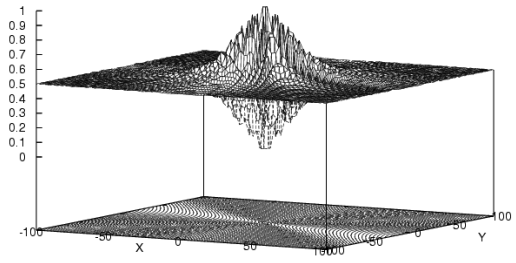
- Strongly multimodal
- Significant interaction between its variables

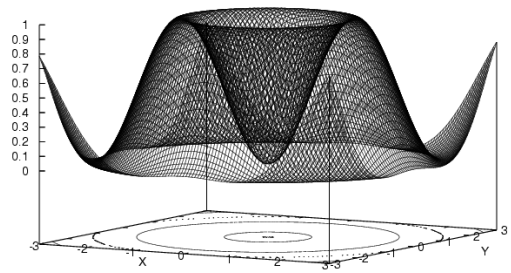


# The Schaffer Function $f_7$

- Designed to trick optimization algorithms
- Many local optima arranged in concentric circles



# Functions

$$
\begin{aligned}
f_1(x) &= \sum_{i=1}^{N} x_i^2 \\
f_2(x) &= \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2 \\
f_3(x) &= \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \\
f_4(x) &= \sum_{i=1}^{N} x_i^2 - 10 \cos 2\pi x_i + 10
\end{aligned}
$$

Table: Benchmark functions used for algorithm comparison

# Functions

$$
\begin{aligned}
f_5(x) &= 1 + \frac{1}{4000}\sum_{i=1}^{N}(x_i - 100)^2 - \prod_{i=1}^{N}\cos\frac{x_i - 100}{\sqrt{i}} \\
f_6(x) &= 20 + e - 20\,e^{0.2\sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}}} - e^{\frac{\sum_{i=1}^{N}\cos 2\pi x_i}{N}} \\
f_7(x) &= 0.5 + \frac{\sin\sqrt{x_1^2 + x_2^2} - 0.5}{1 + 0.001(x_1^2 + x_2^2)}
\end{aligned}
$$

Table: Benchmark functions used for algorithm comparison

# Function Parameters

| | $f_1, f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ |
|---|---|---|---|---|---|---|
| Bounds | $\pm 100$ | $\pm 30$ | $\pm 5.12$ | $\pm 600$ | $\pm 32$ | $\pm 100$ |
| Dimensions | 30 | 30 | 30 | 30 | 30 | 2 |

Table: Parameters of the benchmark functions