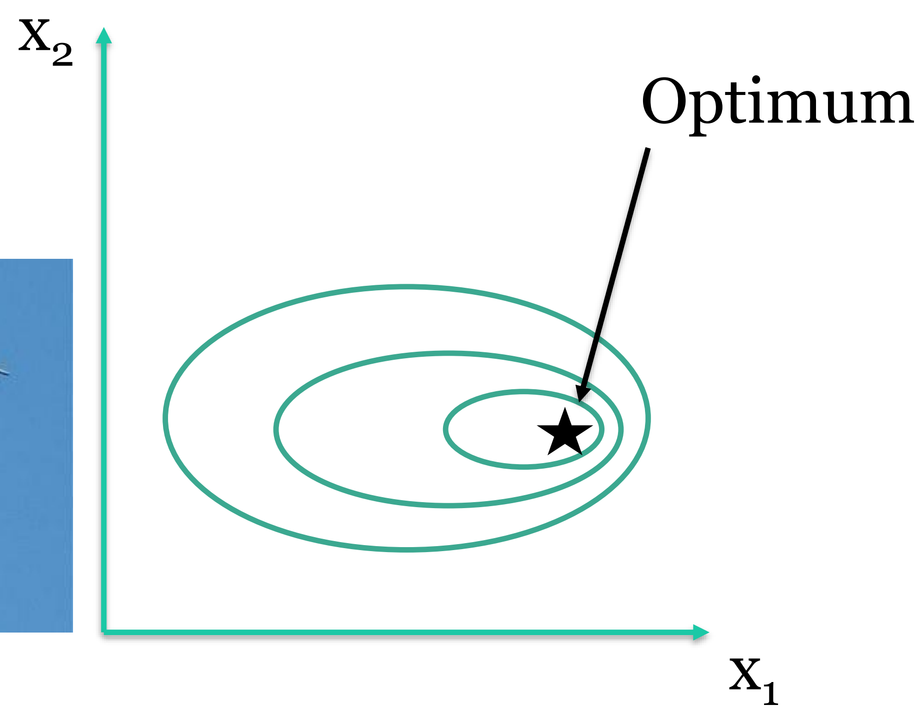
Particle Swarm Optimization

Design Optimization
TMKT48

LINKÖPING UNIVERSITY

Particle Swarm Optimization

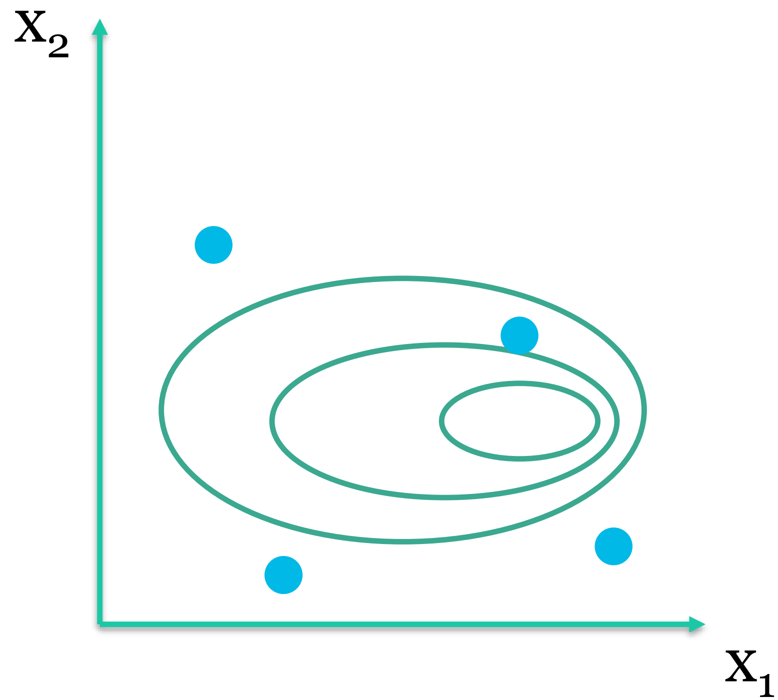
- Eberhart & Kennedy 1995
- Mimics animals that live in swarms / packs
- For example Seagulls



Particle Swarm Optimization

- The algorithm consists of a swarm with a number of individuals that are constant during the optimization

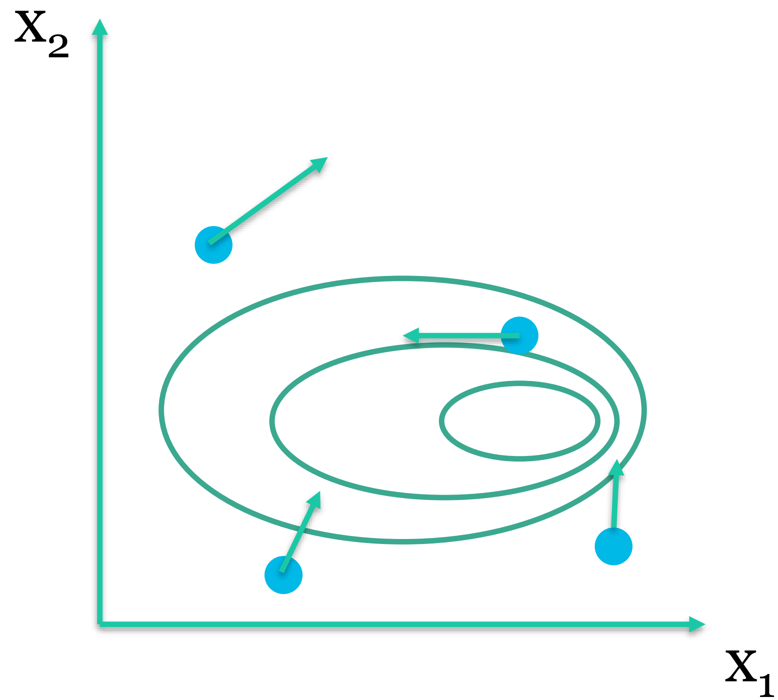
- The individuals start at different locations in the design space



Particle Swarm Optimization

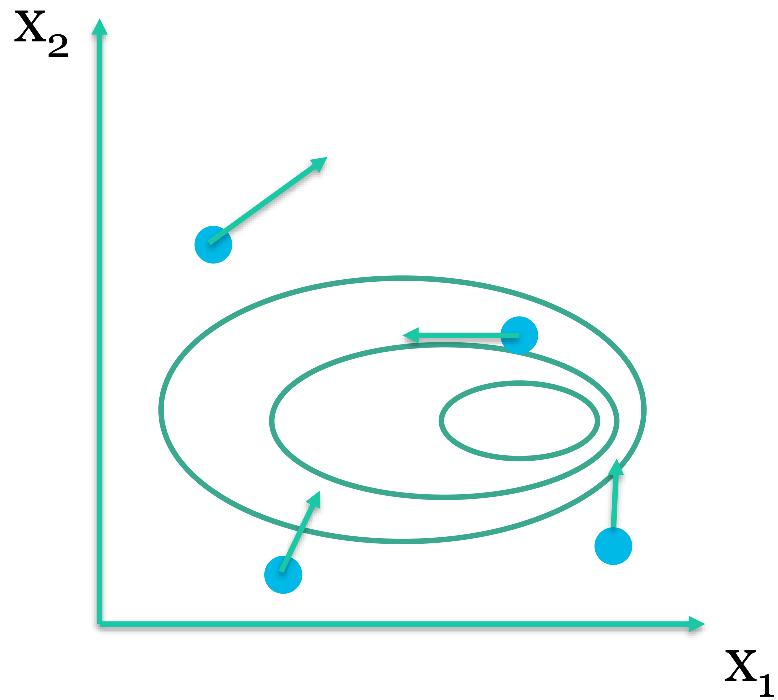
- Each individual is given an initial speed and direction

- The objective function value of each individual is also calculated



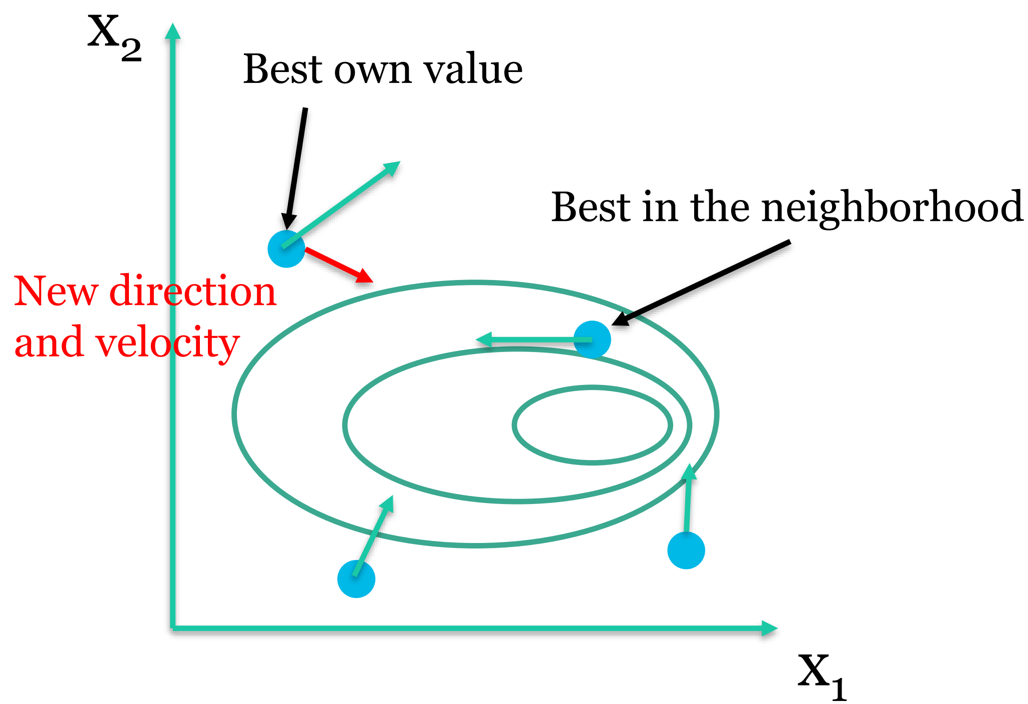
Particle Swarm Optimization

- Each individual will track its best position during the optimization



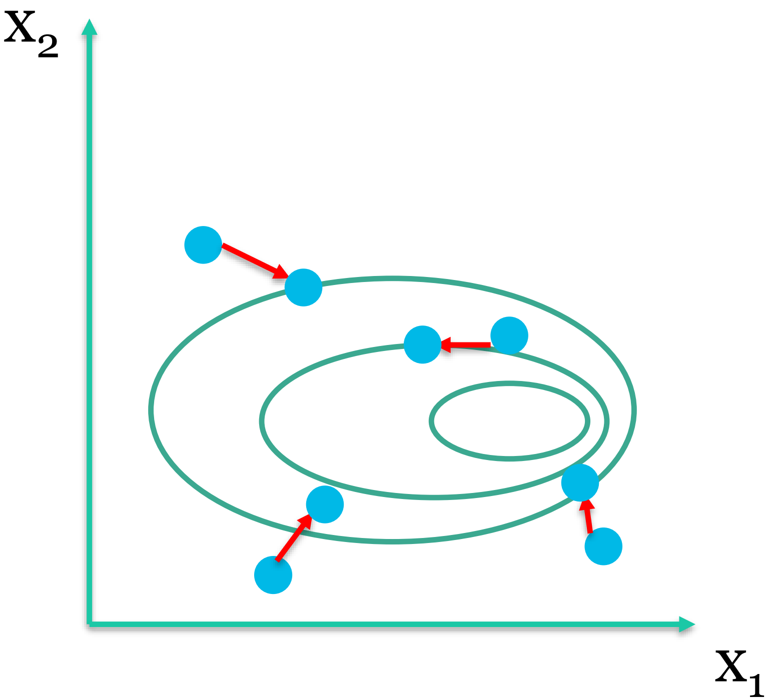
Particle Swarm Optimization

- The new velocity and direction will be a combination of
  - The previous velocity and direction
  - The best position the individual has visited
  - The best position that any neighboring individual has found



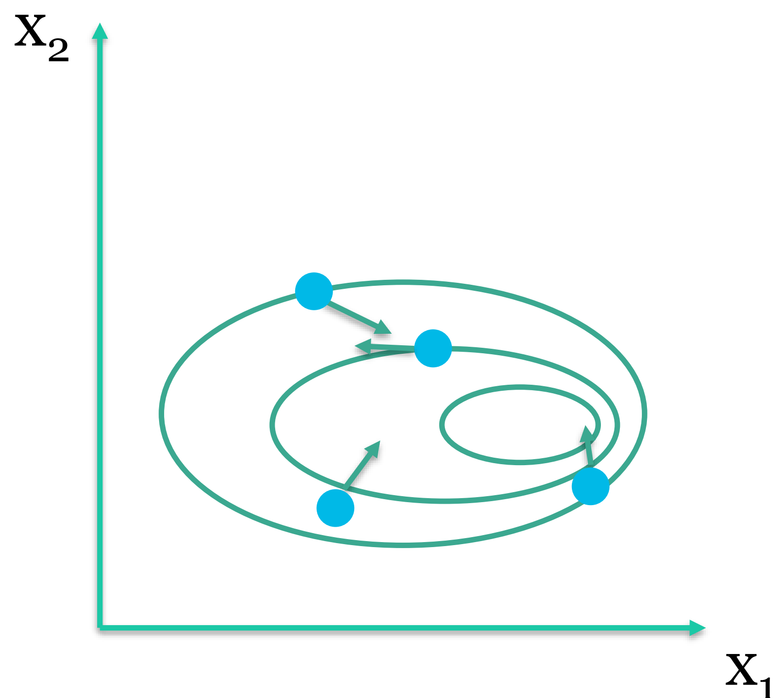
Particle Swarm Optimization

- Move all individuals to their new locations
- Evaluate their objective function values
- Update their best locations found

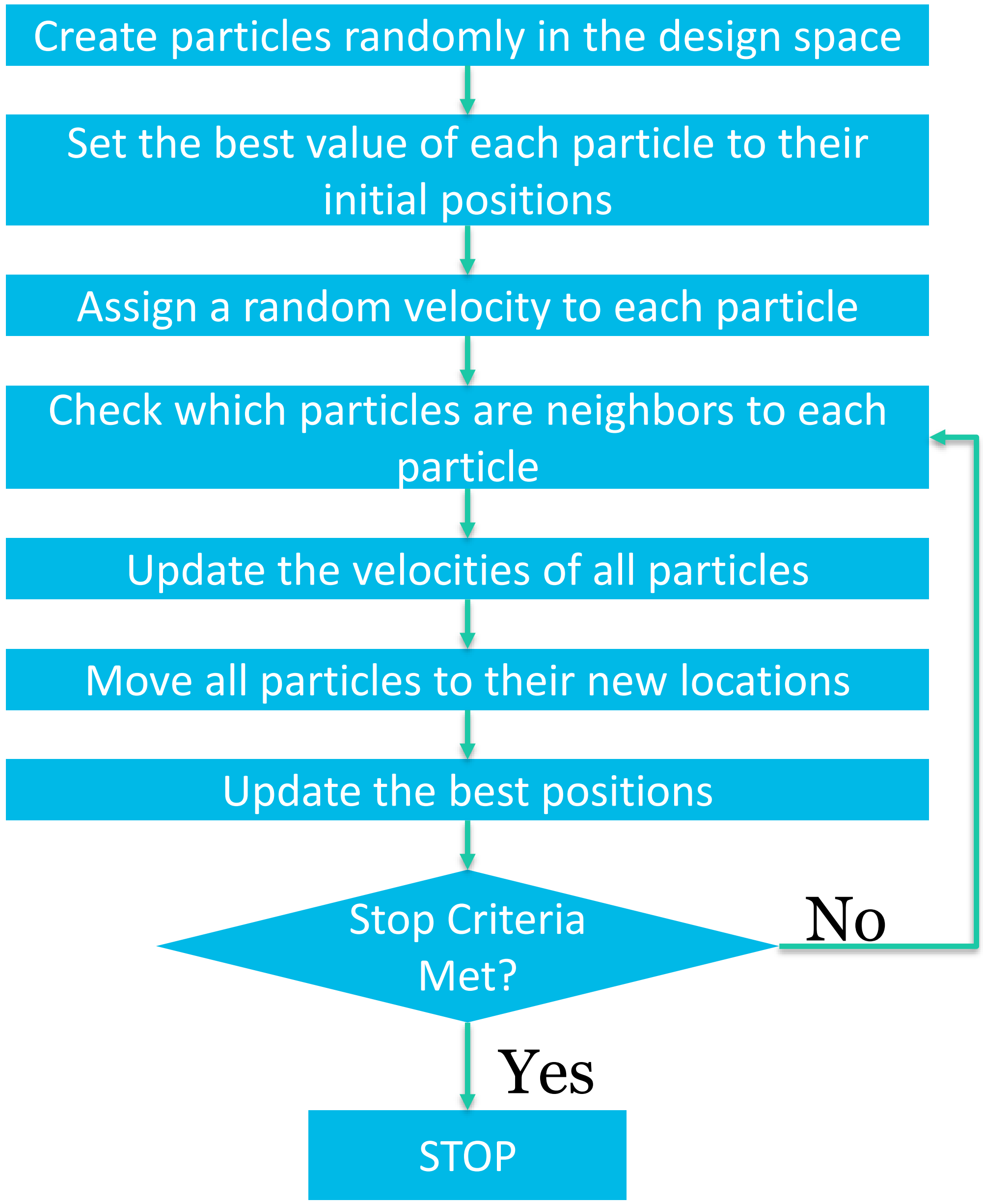


Particle Swarm Optimization

- The individuals will slowly move around towards the optimum until a stop criterion is met
  - No improvement in objective function value
  - Maximum number of evaluations



Algorithm Outline



Question

- How are the number of function evaluations calculated?

- The number of particles multiplied by the number of iterations/movements

Algorithm Outline

- Step 0: Create particles randomly in the design space
- Step 1: Set p(i)=the initial position of particle i
- Step 2: Give each particle an initial speed and direction

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

Algorithm Outline

- Step 3: Select neighbors
- Step 4: Set g(i)=the position of the best neighbor

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

Algorithm Outline

- Step 5: Update the velocity of each particle

$$v = \lambda_1 v + \lambda_2 u_1(p - x) + \lambda_3 u_2(g\text{-}x)$$

- Step 6: Update the position of each particle

$$x = x + v$$

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

Algorithm Outline

- Step 7: Move all points that are outside of the variable limits back into the design space
- Step 8: Evaluate the objective function value of each particle

$$f(i) = f(x(i))$$

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

Algorithm Outline

- Step 9: Check if any new best points were found. If so – update the corresponding variable
  - b
  - d
  - p(i)

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

LINKÖPING UNIVERSITY

Algorithm Outline

- Step 10: Stop criteria met?
  - Otherwise go to Step 2

- λ_j=constants
- b=the best function value overall
- d=the location of the best point
- f(i)=the current objective function value of particle i
- g(i)=the location of the best point in the neighborhood of particle i
- i = particle number
- p(i)=the best position that particle i has visited
- u_j=random numbers between 0 and 1
- v(i)=the velocity of particle i
- x(i)=the current position of particle i

Neighborhood (Swarm Topology)

- Infinite neighborhood
- The m closest neighbors
- Everyone within a certain distance in the design space
- MATLAB: m particles chosen at random

Exploration VS Exploitiation

- Balance between
  - Exploration (find better areas)
  - Exploitation (go towards current best)

- Too much exploration –> Very slow convergence
- Too much exploitation -> Converges fast, but might be to local optimum

PSO VS GA

- Large neighborhood
  - High exploitation
- Trust particles' own history much
  - Medium effect on exploration/exploitation
- Continue in current direction
  - High exploration

$$v = \lambda_1 v + \lambda_2 u_1 (p - x) + \lambda_3 u_2 (g\text{-}x)$$

- High mutation gives
  - High exploration
  - Slow convergence
- Crossover mechanism
- Parent Selection
- Generation Gap

Questions?