

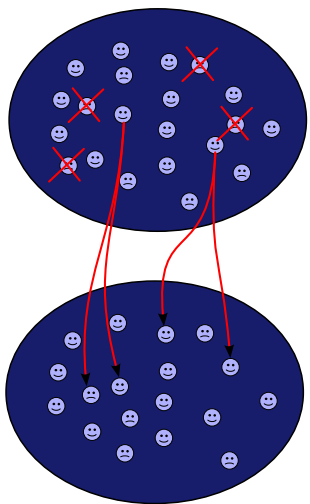
# Genetiska Algoritmer

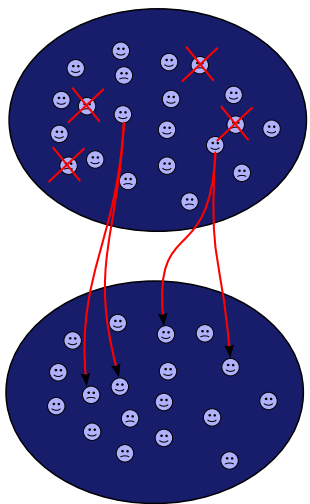
- 1 Grundläggande Idéer
- 2 Algoritmens komponenter
  - Kodning av hypoteser
  - Fitness-funktioner
  - Urvalsmekanismer
  - Skapa variation
- 3 Numerisk optimering
- 4 Genetisk Programmering
  - Exempel

## Parallell optimering inspirerad av biologisk evolution

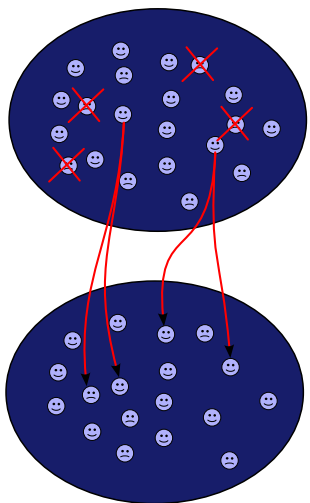
## Parallell optimering inspirerad av biologisk evolution

- Population av hypoteser
- Urvalprocess
- Lokal variation

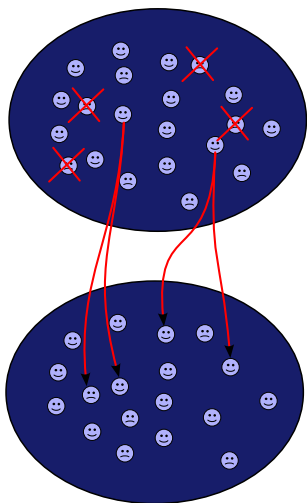




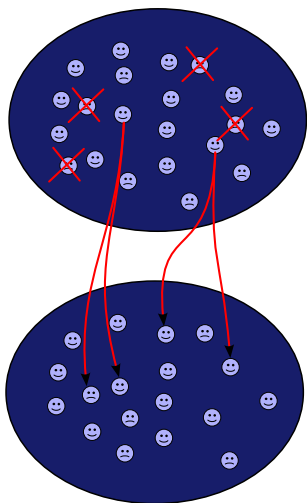
- Population av **individer**



- Population av **individer**
- **Urval** av de bäst fungerande individerna



- Population av **individer**
- **Urval** av de bäst fungerande individerna
- **Variation** skapar nya individer



- Population av **individer**
- **Urval** av de bäst fungerande individerna
- **Variation** skapar nya individer
- Nya **generationer** skapas iterativt

- 1 Grundläggande Idéer
- 2 Algoritmens komponenter
  - Kodning av hypoteser
  - Fitness-funktioner
  - Urvalsmekanismer
  - Skapa variation
- 3 Numerisk optimering
- 4 Genetisk Programmering
  - Exempel

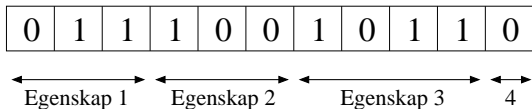
Hur kodas hypoteserna?

Hur kodas hypoteserna?

**Kromosomer** — Binära strängar

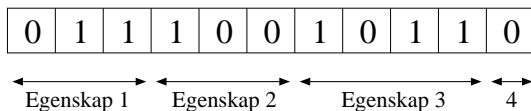
Hur kodas hypoteserna?

**Kromosomer** — Binära strängar



Hur kodas hypoteserna?

**Kromosomer** — Binära strängar



- Genotyp  
Den faktiska lagringen (kromosomerna)
- Fenotyp  
Individens egenskaper (tolkningen)

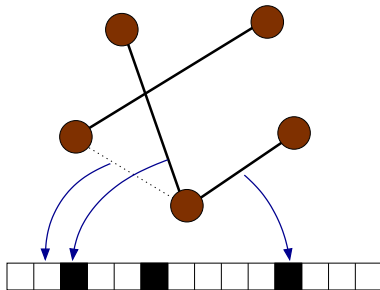
Exempel: Optimalt val av kanter i en graf

Exempel: Optimalt val av kanter i en graf

Kanterna kodas som en bit-sträng

Exempel: Optimalt val av kanter i en graf

Kanterna kodas som en bit-sträng



Måste allt lagras som bit-strängar?

Måste allt lagras som bit-strängar?

Varianter:

- Andra heltal än 0/1

Måste allt lagras som bit-strängar?

Varianter:

- Andra heltal än 0/1
- Reella tal

Måste allt lagras som bit-strängar?

Varianter:

- Andra heltal än 0/1
- Reella tal
- Variabel längd

Måste allt lagras som bit-strängar?

Varianter:

- Andra heltal än 0/1
- Reella tal
- Variabel längd
- Trädstrukturer

## Fitness-funktionen

Mått på hur bra hypotesen är

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

- Total väglängd i en graf

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

- Total väglängd i en graf
- Felet vid funktionsanpassning

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

- Total väglängd i en graf
- Felet vid funktionsanpassning
- Prestanda för en simulerad robot

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

- Total väglängd i en graf
- Felet vid funktionsanpassning
- Prestanda för en simulerad robot
- Antal vunna partier (spel)

## Fitness-funktionen

Mått på hur bra hypotesen är

$$f : \text{kromosom} \mapsto \mathcal{R}$$

Exempel:

- Total väglängd i en graf
- Felet vid funktionsanpassning
- Prestanda för en simulerad robot
- Antal vunna partier (spel)

Evalueringen av fitnessfunktionen är ofta den *beräkningskrävade delen* av en genetisk algoritm

# Urval

## Urval

Grundidé: Behåll individer med hög fitness

## Urval

Grundidé: Behåll individer med hög fitness

- **Roulette selection**

Sannolikheten att överleva proportionell mot  $f$

## Urval

Grundidé: Behåll individer med hög fitness

- **Roulette selection**  
Sannolikheten att överleva proportionell mot  $f$
- **Ranking selection**  
Urval baserat på ordningsnummer istället för fitness-värde.

## Urval

Grundidé: Behåll individer med hög fitness

- **Roulette selection**  
Sannolikheten att överleva proportionell mot  $f$
- **Ranking selection**  
Urval baserat på ordningsnummer istället för fitness-värde.
- **Tournament selection**  
Slumpmässiga par bildas och den med högst fitness överlever

## Urval

Grundidé: Behåll individer med hög fitness

- **Roulette selection**  
Sannolikheten att överleva proportionell mot  $f$
- **Ranking selection**  
Urval baserat på ordningsnummer istället för fitness-värde.
- **Tournament selection**  
Slumpmässiga par bildas och den med högst fitness överlever
- **Elitism**  
Garanterad överlevnad för de bäst rankade individerna

- **Mutationer**  
Små slumpmässiga ändringar
- **Korsningar**  
Blandning av individernas egenskaper

# Mutationer

## Mutationer

- Gör slumpmässiga ändringar i kromosomerna

## Mutationer

- Gör slumpmässiga ändringar i kromosomerna
- Vald kodning spelar stor roll

# Korsningar

## Korsningar

- Välj två individer med hög fitness
- Byt delar av kromosomerna med varandra

## Korsningar

- Välj två individer med hög fitness
- Byt delar av kromosomerna med varandra

### One-point crossover

## Korsningar

- Välj två individer med hög fitness
- Byt delar av kromosomerna med varandra

One-point crossover

Multi-point crossover

## Tillämpning på vanliga optimeringsproblem

Tillämpning på vanliga optimeringsproblem

Antag att vi söker  $\max f(x, y)$

Tillämpning på vanliga optimeringsproblem

Antag att vi söker  $\max f(x, y)$

Kodning: kromosom med två reella tal

Tillämpning på vanliga optimeringsproblem

Antag att vi söker  $\max f(x, y)$

Kodning: kromosom med två reella tal

Varje individ är en punkt i planet

Tillämpning på vanliga optimeringsproblem

Antag att vi söker  $\max f(x, y)$

Kodning: kromosom med två reella tal

Varje individ är en punkt i planet

- Mutation  
Spridning parallellt med koordinataxlarna

Tillämpning på vanliga optimeringsproblem

Antag att vi söker  $\max f(x, y)$

Kodning: kromosom med två reella tal

Varje individ är en punkt i planet

- Mutation  
Spridning parallellt med koordinataxlarna
- Korsning  
Nya punker med  $x$  från en förälder och  $y$  från en annan

## Exempel: Optimerad kodgenerering vid kompilering

Exempel: Optimerad kodgenerering vid kompilering

ACOVEA — Analysis of Compiler Options via Evolutionary Algorithms

Exempel: Optimerad kodgenerering vid kompilering

ACOVEA — Analysis of Compiler Options via Evolutionary Algorithms

System för att hitta de optimala kompilatorinställningarna för ett givet C-program

# Genetisk programmering

# Genetisk programmering

Användning av GA för att skapa program automatiskt

# Genetisk programmering

Användning av GA för att skapa program automatiskt

- Hur representeras programmen?

# Genetisk programmering

## Användning av GA för att skapa program automatiskt

- Hur representeras programmen?
- Hur mäter man fitness?

# Genetisk programmering

## Användning av GA för att skapa program automatiskt

- Hur representeras programmen?
- Hur mäter man fitness?
- Hur gör man mutationer?

# Genetisk programmering

## Användning av GA för att skapa program automatiskt

- Hur representeras programmen?
- Hur mäter man fitness?
- Hur gör man mutationer?
- Hur gör man korsningar?

## Representation av programmen

Representation av programmen

Vanliga språk är inte lämpliga

Representation av programmen

Vanliga språk är inte lämpliga

- Träd med operatorer

## Representation av programmen

Vanliga språk är inte lämpliga

- Träd med operatorer
- Instruktionslista

# Exempel

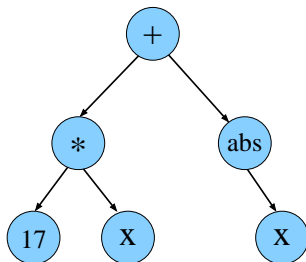
## Funktionsanpassning

## Exempel

Exempel

Funktionsanpassning

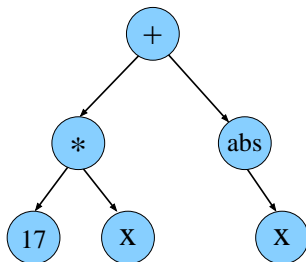
Representation av programmet



# Exempel

## Funktionsanpassning

### Representation av programmet

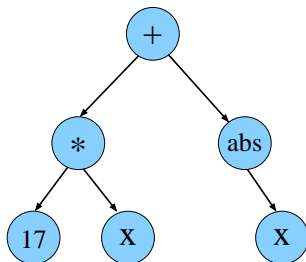


- Mutationer

## Exempel

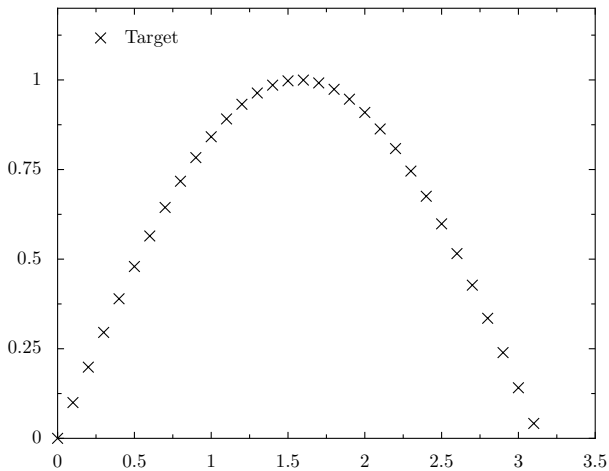
### Funktionsanpassning

### Representation av programmet

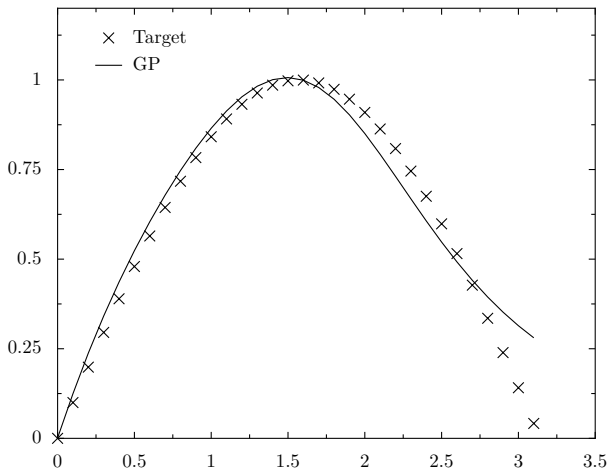


- Mutationer
- Korsningar

## Målfunktion

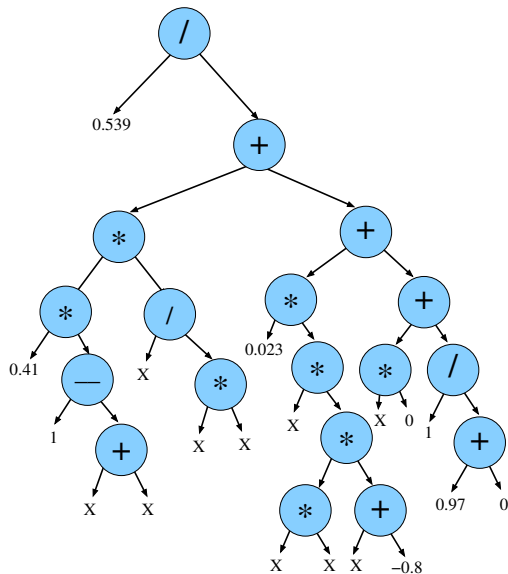


## Funnen lösning





## Exempel



## Bloating

Ansamlandet av  
onödiga delar i  
kromosomer med  
variabel längd