UiO : **Department of Informatics**
University of Oslo

Kyrre Glette – kyrrehg@ifi

INF3490 – Evolvable Hardware

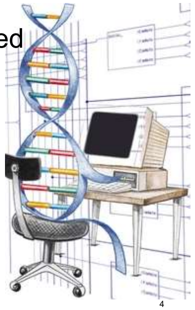
Cartesian Genetic Programming

Overview

- Introduction to Evolvable Hardware (EHW)
- Cartesian Genetic Programming
- Applications of EHW

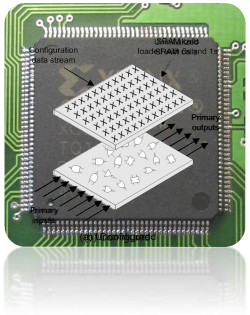
Evolvable Hardware (EHW)

- Hardware systems designed/ modified automatically by EAs
- A string of symbols/bits is evolved by an EA and translated into a HW system
- Offline EHW
  - Solutions are simulated in a PC
- Online EHW
  - Solutions are tested on target HW



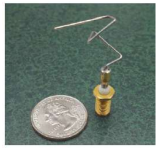
EHW

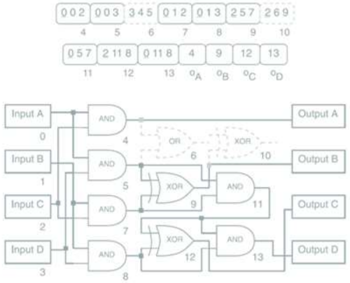
- FPGA
  - Reconfigurable hardware chip
  - Useful for online EHW
- On-chip evolution
  - EA running on the target chip, together with solutions
- Run-time adaptable EHW
  - Evolution can modify the system during operation



Applications of EHW

- Pattern recognition / classification circuits
- Digital image filters
- Evolution of analog circuits
- Cache mapping functions
- On-the-fly compression for printers
- Spacecraft antenna



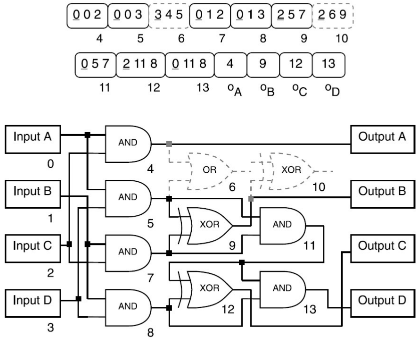


CARTESIAN GENETIC PROGRAMMING

Cartesian Genetic Programming (CGP)

- A type of Genetic Programming
- Allows restrictions compared to general GP:
  - Integer genome
  - Tree nodes are mapped to a grid
  - Connectivity can be restricted
- Popular in Evolvable Hardware applications
  - But can be used for many other things as well

Example structure: Digital circuit



CGP genome

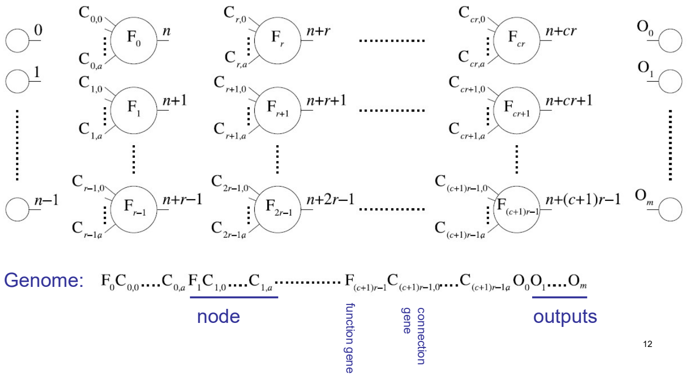
- Internal node genes:
  - Node type: index to lookup table of functions
  - Inputs: index of other nodes
  - Optional: additional parameters
- Output node gene:
  - Internal node index

CGP parameters

- Columns: n_c
- Rows: n_r
- Levels-back: l
  - How many of the previous columns a node can connect to
- Columns x rows defines the maximum number of nodes in the graph

---

General structure



Genome: $F_0C_{0,0}\ldots C_{0,a}F_1C_{1,0}\ldots C_{1,a}\ldots\ldots F_{(c+1)r-1}C_{(c+1)r-1,0}\ldots C_{(c+1)r-1a}O_0O_1\ldots O_m$

Advantages of CGP

- Easy implementation
  - Fixed genome size and simple representation
  - Simple mutation and crossover
- Bloat is restricted
  - The number of nodes is restricted
- Regular structure suitable for e.g. hardware implementation
  - A grid structure with limited connectivity ideal for HW routing

Other features of CGP

- Reuse of parts of the tree is possible
- Allows multiple outputs
- Parts of the genome may be non-coding
  - This has an analogy in biology, where only a fraction of the DNA is composed of *exons* (“coding” genes).
  - The other part is called *introns* (non-coding genes, sometimes called “junk” DNA). It is however believed that these are useful for something.
  - Likewise, the genetic redundancy (neutrality) in CGP is thought to be positive for the evolutionary search.

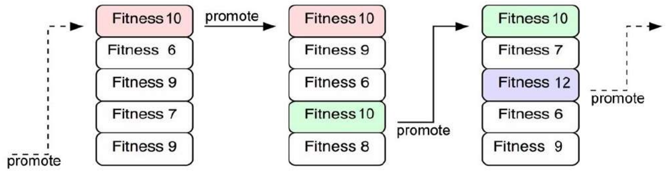
Genetic operations in CGP

- Mutation
  - Select randomly a number of genes to mutate
  - Change to new (valid) random values
- Crossover
  - One-point crossover or other variants directly on the genome
- Usually only mutations are used
  - Many applications find crossover to have a destructive effect - it disrupts the tree structure too much

---

Evolution in CGP

- The most popular is a variant of ES called (1+4) ES
- Choose children which have >= fitness than parent

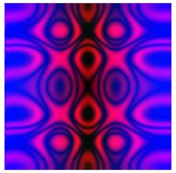


CGP can code:

- Circuits
- Mathematical functions / equations
- Neural networks
- Programs
- Machine learning structures
- …

UiO : **Department of Informatics**
University of Oslo

Example: Art



- Inputs: image pixel position x,y
- Outputs: r,g,b intensities per pixel
    - Or single monochrome intensity

```
r = f1(x,y)
g = f2(x,y)
b = f3(x,y)
```

Function gene	Function definition
0	x
1	y
2	$\sqrt{x+y}$
3	$\sqrt{\mid x-y \mid}$
4	$255(\mid \sin(\frac{2\pi}{255}x) + \cos(\frac{2\pi}{255}y) \mid)/2$
5	$255(\mid \cos(\frac{2\pi}{255}x) + \sin(\frac{2\pi}{255}y) \mid)/2$
6	$255(\mid \cos(\frac{3\pi}{255}x) + \sin(\frac{2\pi}{255}y) \mid)/2$
7	$\exp(x+y) \pmod{256}$
8	$\mid \sinh(x+y) \mid \pmod{256}$

UiO : **Department of Informatics**
University of Oslo

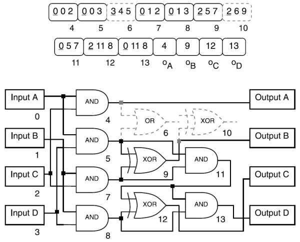
Example: Evolvable Hardware 1

- Evolution of combinational circuit, e.g. multiplier
- 2-bit multiplier:

  2x2=4 inputs

  4 outputs
- Fitness:

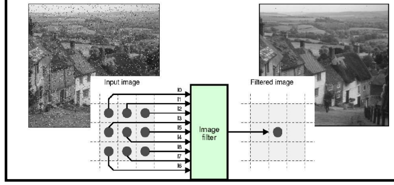
  # correct output combinations (of 16)



UiO : **Department of Informatics**
University of Oslo

Example: Evolvable Hardware 2

- Evolution of digital image filters
- Input: distorted image
- Output: filtered image
- Fitness: distance between filtered and original image



Number	Function	Description
0	$x \vee y$	binary or
1	$x \wedge y$	binary and
2	$x \oplus y$	binary xor
3	$x + y$	addition
4	$x +^s y$	addition with saturation
5	$(x+y) >> 1$	average
6	$Max(x,y)$	maximum
7	$Min(x,y)$	minimum

UiO : **Department of Informatics**
University of Oslo

Example result (Slide from Sekanina)

a) Image corrupted by 5% salt-and-pepper noise

PSNR: 18.43 dB (peak signal to noise ratio)

b) Original image

c) Median filter (kernel 3x3)

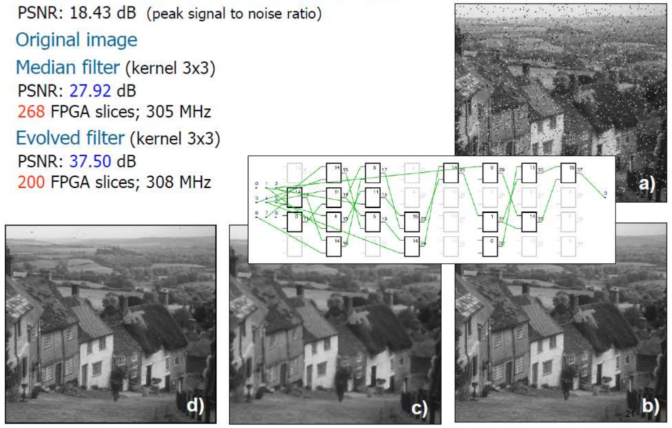
PSNR: 27.92 dB

268 FPGA slices; 305 MHz

d) Evolved filter (kernel 3x3)

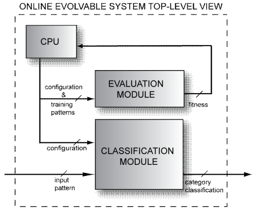
PSNR: 37.50 dB

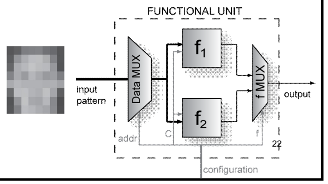
200 FPGA slices; 308 MHz



Example: EHW 3 (ROBIN group)

- Evolution of HW classifiers
  - Input: signal to be classified
  - Output: classification result
- Fitness:
  - # correctly classified training samples
- Ensemble model
- On-chip system





Challenges of EHW

- Scalability – It's hard to evolve large systems!
  - General challenge in EC
  - Evolution of larger combinational circuits is difficult
    - Large *and* difficult search space
    - Time-consuming fitness function
    - 4x4 multiplier is hard
- On-chip evolution
  - Less flexibility offered by HW
  - Reconfiguration can be challenging

UiO : **Department of Informatics**
University of Oslo

Kyrre Glette – kyrrehg@ifi

INF3490 – Swarm Intelligence
Particle Swarm Optimization

UiO : **Department of Informatics**
University of Oslo

Overview

- Introduction to swarm intelligence principles
- Particle Swarm Optimization (in depth)
- Stigmergy, pheromones, and ACO (briefly)
- Swarm robotics (briefly)

UiO : **Department of Informatics**
University of Oslo

Swarms in nature

http://youtu.be/kdECYXdW9Tc

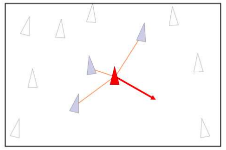
UiO : **Department of Informatics**
University of Oslo

Fish, birds, ants, termites, …

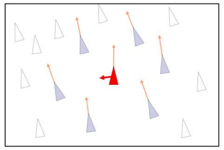
Key features

- Simple local rules
- Local interaction
- Decentralized control
- Complex global behavior
  - Difficult to predict from observing the local rules
  - *Emergent* behavior

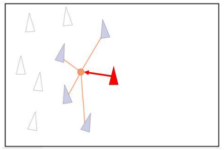
Flocking model – "boids"



Separation – avoid crowding



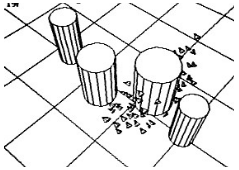
Alignment – steer towards average heading

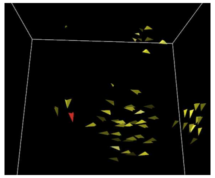


Cohesion – steer towards average position

Only considering the boid's neighborhood

Result - boids





Original: http://youtu.be/86iQiV3-3IA

Netlogo: "Flocking 3D Alternate" model

Application: Computer graphics

https://youtu.be/UM8SzF6_0sM

https://youtu.be/ttxXH4WkrJM?t=1m46s

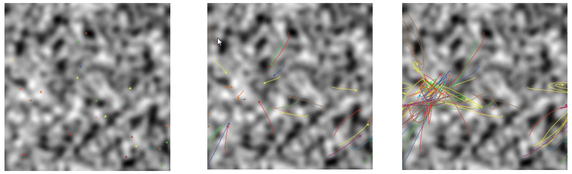
Applications in bio-inspired computing

- **Particle swarm optimization**
  - Parameter optimization
- Ant colony optimization
  - Graph-based optimization problems (e.g. TSP)
- Artificial immune systems
  - Classification, anomaly detection
- Swarm robotics
  - Achieve complex behavior in robotic swarms through simple local rules

---

Particle Swarm Optimization (PSO)

- Optimizes a population of solutions
  - A *swarm* of *particles*



Principle

- Evaluate your present position
- Compare it to your previous best and neighborhood best
- Imitate self and others

Simplified PSO algorithm

- For each particle *i* in the swarm
  - Calculate fitness
  - Update local best
  - Find neighborhood best
  - Update velocity
  - Update position

PSO update formulas

For each dimension *d* in particle *i*:

1. Velocity update

$$v_{id}^{(t+1)} \leftarrow \underbrace{\alpha v_{id}^{(t)}}_{\text{inertia}} + \overset{\text{random}}{U(0,\beta)} \underbrace{\left(p_{id} - x_{id}^{(t)}\right)}_{\text{direction personal best}} + \overset{\text{random}}{U(0,\beta)} \underbrace{\left(p_{gd} - x_{id}^{(t)}\right)}_{\text{direction neighborhood best}}$$

2. Position update

$$x_{id}^{(t+1)} \leftarrow x_{id}^{(t)} + v_{id}^{(t+1)}$$

What happens?

- A particle circles around in a region centered between the bests of itself and its neighbors
- The bests are updated and the particles cluster around better regions in the search space
- The way good solutions are propagated depends on how we define the neighborhood

Neighborhood topologies

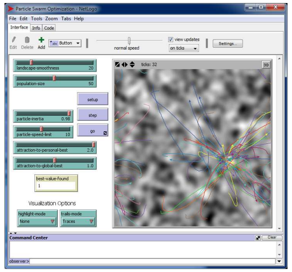
- *gbest*: all particles are connected
  - Every particle gets information about the global best value
  - Can converge (too) fast
- *lbest*: connected to *K* nearest neighbors in a wrapped population array
  - Slower convergence, depending on *K*
  - More areas are searched in parallel
- Several other topologies exist

PSO parameters

- Particle:
  - Usually a D-dimensional vector of real values
  - Binary variant exists
- Swarm size: usually 10 < N < 100
- Recommended α = 0.7298
- Recommended β = 1.4961

Parameter experimentation

- NetLogo
  - Particle Swarm Optimization model
- Model uses *gbest* neighborhood
- Download and try



☑ view updates
on ticks
ticks: 32
landscape-smoothness 20
population-size 50
particle-inertia 0.98
particle-speed-limit 10
attraction-to-personal-best 2.0
attraction-to-global-best 1.0
best-value-found 1

Visualization Options
highlight-mode None
trails-mode Traces
observer>

Advantages of PSO

- Few parameters
- Gradient free
- Decentralized control (depends on variant.)
- Simple to understand basic principle
- Simple to implement

PSO vs. Evolutionary Algorithms

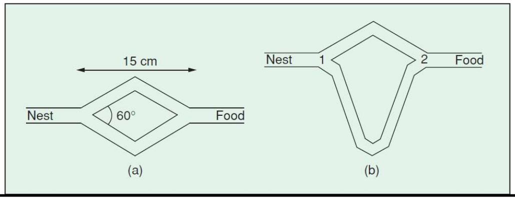
- Both are population based
- PSO: No selection – all particles survive
- Information exchange between solutions:
  - PSO: neighborhood best
  - GA: crossover (and selection)

PSO applications

- Similar application areas as EAs
  - Most optimization problems
- Image and video analysis
- Electricity network optimization
- Neural networks
- …

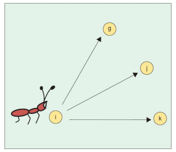
Stigmergy

- Ants deposit a substance called *pheromone* when walking to and from food sources
  - Other ants can sense this and follow the same path
- This kind of communication through the environment is called *stigmergy*



Ant Colony Optimisation (ACO)

- Inspired by ants' use of pheromones
- Ants construct solutions in a graph
  - Probability of choosing a new edge is proportional with its pheromone level
- Pheromone update on edges
  - (Good) solutions deposit pheromones
  - Old pheromones evaporate



ACO applications

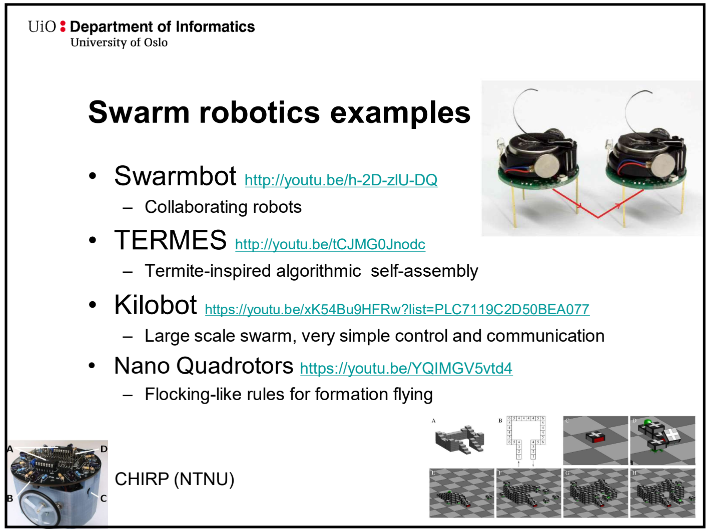
- Telecomunication networks
- Scheduling problems
- Vehicle routing (truck fleet)
- ...

- Further reading:
  M. Dorigo et al, *Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique*,
  IEEE Computational Intelligence Magazine, Nov. 2006

Swarm robotics

- Inspired by nature's swarms
  - Simple rules
  - Local interaction
  - Decentralized control
  - Complex global behavior (soon?)
- Advantages
  - Cheap components
  - No single point of failure
  - Many configurations possible
- Possible applications
  - Search and rescue
  - Remote area exploration
  - Construction

https://youtu.be/NDjTqQ7xbWQ



UiO : **Department of Informatics**
University of Oslo

Swarm robotics examples

- Swarmbot http://youtu.be/h-2D-zlU-DQ
    - Collaborating robots
- TERMES http://youtu.be/tCJMG0Jnodc
    - Termite-inspired algorithmic self-assembly
- Kilobot https://youtu.be/xK54Bu9HFRw?list=PLC7119C2D50BEA077
    - Large scale swarm, very simple control and communication
- Nano Quadrotors https://youtu.be/YQIMGV5vtd4
    - Flocking-like rules for formation flying

CHIRP (NTNU)