



Vaasan yliopisto
UNIVERSITY OF VAASA

Telecommunications seminar

“Evolutionary algorithms in Communications and systems”

Introduction lecture II: More about EAs

Timo Mantere

Professor
Communications and systems engineering
University of Vaasa

10.3.2015




UNIVERSITY of VAASA
*Communications and Systems
Engineering Group*

Evolutionary algorithms in communications

MORE ABOUT Eas – This time we introduce different EAs

- Differential evolution
- Swarm intelligence
- Cultural algorithms
- Meta-EA
- Island models
- Cellular EA
- Co-evolution
- Pareto front
- Hybridization

We will go to the applications in the exercises



UNIVERSITY of VAASA
*Communications and Systems
Engineering Group*

Application areas

- Evolutionary algorithms have been applied to almost all possible search, design and optimization problems that anyone can think of
 - Just take a look of IEEE database with keywords: 'whatever' + "genetic algorithm" or "evolutionary algorithm"
<http://ieeexplore.ieee.org/search/advsearch.jsp>
- They are particularly strong with problems that do not at be solved with mathematical optimization methods.
 - Nonlinear problems
 - NP-complete problems
 - Noncontinuous problems
 - The problems that cannot be expressed mathematically
 - ☞ It is enough that we can evaluate the fitness value somehow, and the evaluator can be a human being, simulator, computer program etc.



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Simple GA

In the original GA by John Holland

- Use binary coding
- Random initial population
- One-point, or multi-point crossover
- Crossover more important than mutation
- The roulette wheel parent selection, each individual had a chance to become parent that was linearly dependent of it's fitness value => each individual got as big share of the roulette wheel as it's fitness value compared to the sum of fitness values of all individuals.
- Mutation: bit flipping with constant probability
- All old individuals are replaced, so there was no elitism

Nowadays this Hollands original GA is called as Simple GA (SGA). It is rarely used any more, mainly it can be used as comparison level, when the new evolution algorithms are tested



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Differential evolution (DE)

- The previous slides showed that arithmetic crossover, which is quite often used in the floating point coded GAs is problematic, since it is averaging method and it losses genetic information
- In floating point GAs this problem is compensated with Gaussian distributed mutation, but these usually takes a lot of time to provide exact solutions (finding the last decimals takes a long time)
- Both of these problems are avoided by using DE instead of floating point GAs



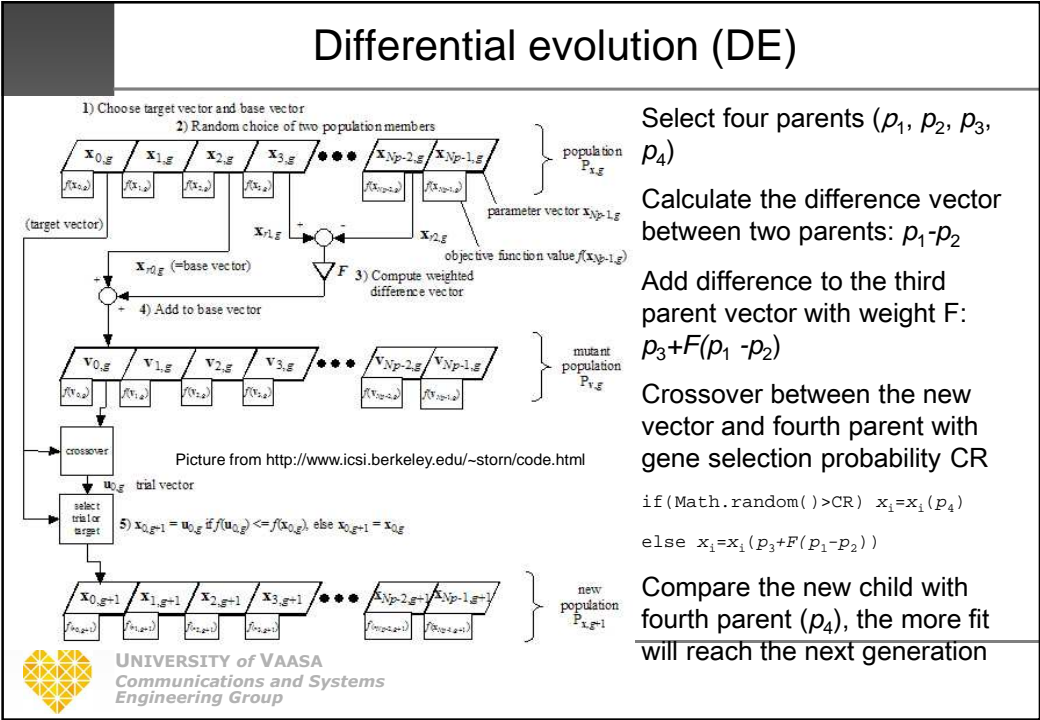
UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Differential evolution (DE)

- Combined crossover-mutation operation of DE can lead new gene value also to be higher or lower than parent gene value
 - DE also requires the handling of limit violations, since the differential can move the gene value out of legal range
- DE finds last decimals very effectively because the differentials between parents becomes smaller and smaller when the population diversity decreases
- Obviously if there is too fast convergence, the DE system can also get stuck to the local optimum
 - This can be avoided by several different approaches, mainly additional mutations, or using DE as hybrid with other algorithms



UNIVERSITY of VAASA
Communications and Systems
Engineering Group



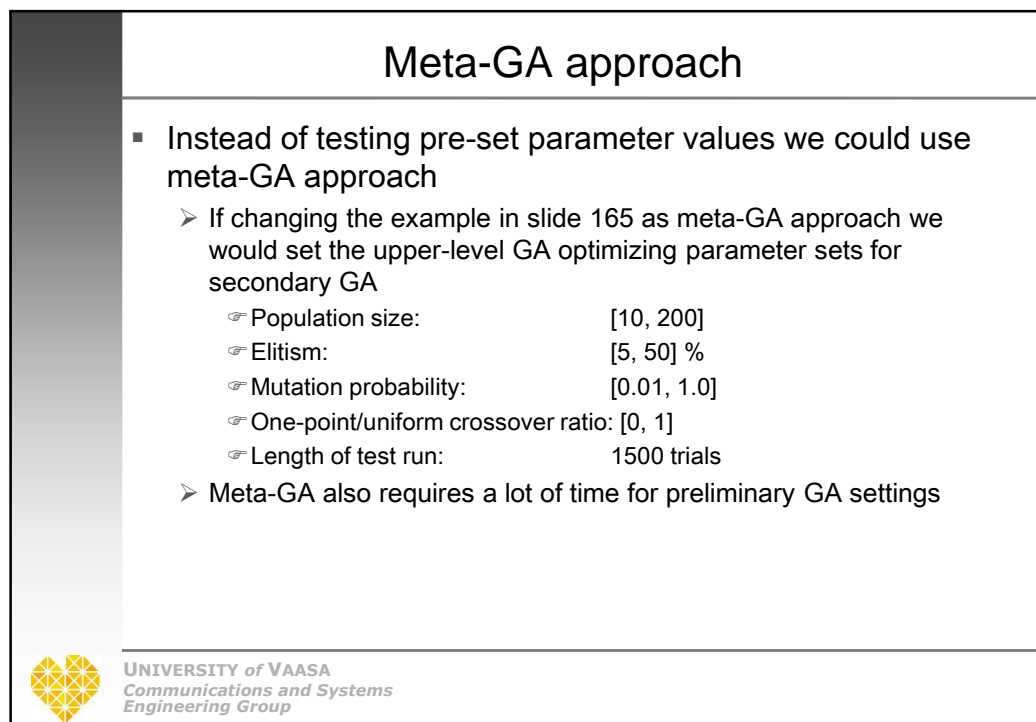
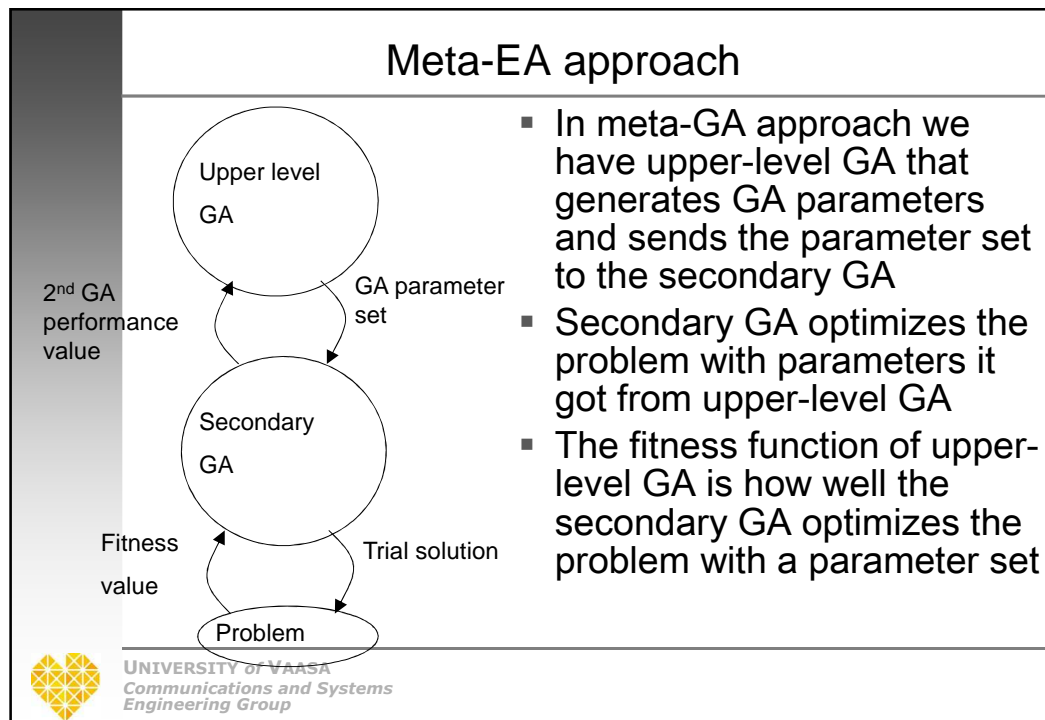
DE operation example

Parent 1:	0.12	0.65	0.45	0.32	0.77
Parent 2:	0.16	0.02	0.77	0.34	0.31
Difference:	-0.04	0.63	-0.32	-0.02	0.46
*F (=0.5)	-0.02	0.32	-0.16	-0.01	0.23
Parent 3:	0.45	0.33	0.23	0.77	0.81
Donor vector: (=P3+F*diff)	0.43	0.65	0.07	0.76	1.00* (*overflow if values [0, 1])
Parent 4:	0.23	0.77	0.43	0.88	0.96
New child:	0.43	0.77	0.07	0.76	0.96

(CR=0.6 meaning 3/5 of the gene values taken from donor vector and 2/5 from the parent 4)

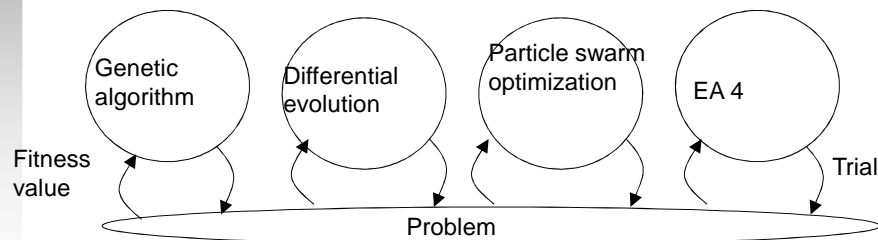
http://en.wikipedia.org/wiki/Differential_evolution

UNIVERSITY of VAASA
Communications and Systems
Engineering Group



Best of Different strategies method

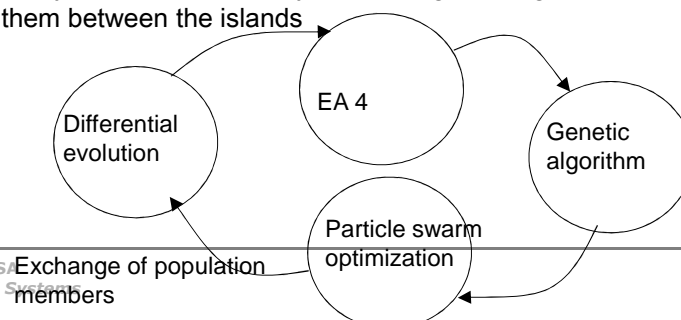
- One possibility is also to optimize the problem with different algorithms
 - All methods could run with their default parameter settings
 - After each one have optimized the problem, we just select the best solution of the method that was found to be the best
 - The problem might be the work needed for coding all the different algorithms



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Different strategies island model

- The island model EA method can also be extended to different strategies
 - Each island uses different optimization method and have its own population
 - Every now and then some population members (solutions) are exchanged between islands
 - This method requires that chromosomes with different methods are coded similarly, or at least are easy to exchange coding when changed them between the islands



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Swarm intelligence

- Sometimes we might think that there exist more intelligence in the problem than the single individuals can learn
 - Some kind of group or Meta-knowledge of the problem in hand
 - This knowledge of the problem can be used with different optimization methods
 - ☞ Group based methods like particle swarm optimization and ant colony optimization that only contain individuals
 - ☞ System that has some central upper knowledge coded somewhere else than just into the individuals -> Cultural algorithms



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Swarm intelligence


- Particle swarm optimization mimes the bird or fish flocking behavior
 - One bird or fish is the leader and the rest of the group follows the leader
 - ☞ The leader can change
 - In particle swarm optimization we have a group of points and after they are evaluated the best one is leader and all the other points moves towards the leader according some step size
 - ☞ The new points are evaluated and if some of them is better than leader, it will become new leader
 - ☞ http://en.wikipedia.org/wiki/Particle_swarm_optimization



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Swarm intelligence

- Ant colony optimization mimes the behavior of ants when they search for food
 - The ants leave the trail of smell in the path when they move
 - The smell in the path where they bring food will get stronger because many ants traverse it
 - ☞ After the food run out the ants start to find new food source and the smell trace will evaporate over time
 - ☞ Usually used as route search algorithm
 - ☞ http://en.wikipedia.org/wiki/Ant_colony_optimization
 - ☞ http://en.wikipedia.org/wiki/Ant_colony




UNIVERSITY of VAASA
Communications and Systems
Engineering Group

ACO, pheromone trails, Sudoku 11.3.2015


In our ant colony algorithm, the initial generation is created randomly. After each generation, the old pheromone paths are weakened by 10%. Then the nine best individuals update the pheromone trails by adding the strength value by

$$\text{Strength} += \begin{cases} \frac{1}{\sqrt{\text{fitnessval}_{ue}}}, & \text{fitnessval}_{ue} > 2 \\ \frac{1}{2}, & \text{fitnessval}_{ue} = 2 \end{cases}$$



for each number that appears in each location (Strength_{*n*}) of that Sudoku solution proposal, i.e., an ant. These strength values were also chosen after several test runs. The system heavily favors the near-optimal solutions.

New ants are generated with a weighted random generator, where each number *n* has a probability $\frac{\text{Strength}_n + \delta}{\sum_{n=1}^9 (\text{Strength}_n + \delta)}$, where $\delta \sim \text{Unif}(0.000001, 0.01)$ to be assigned to each location of the Sudoku solution trial. This means that there is small random change for even those numbers to be drawn that have zero pheromone. Without any random slack the optimization would get stuck in the situation where it cannot create a new generation of unique solutions. Table 1 summarizes properties of our algorithms.



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Cultural algorithms

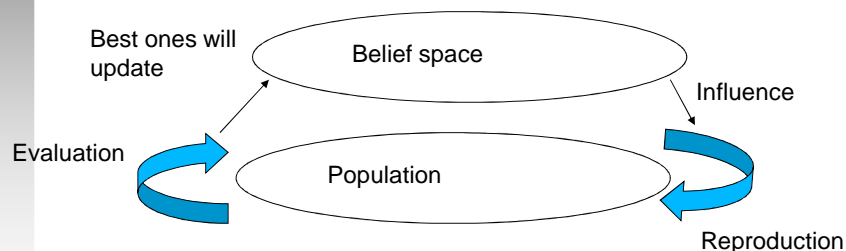
- Particle swarm and ant colony methods are still consisted only individuals and the group intelligence only appears as the individual behavior
- If we add central knowledge to the system we get cultural algorithm
- This central knowledge is usually expressed as a form of belief space
 - Belief space gathers the history data of individuals performance and guides the production of new individuals according the learned cultural knowledge



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Cultural algorithms

- The best individuals will update the belief space
- Belief space influences the reproduction by guiding what kind of new individuals should be generated
 - Some that are good according the learned meta-knowledge

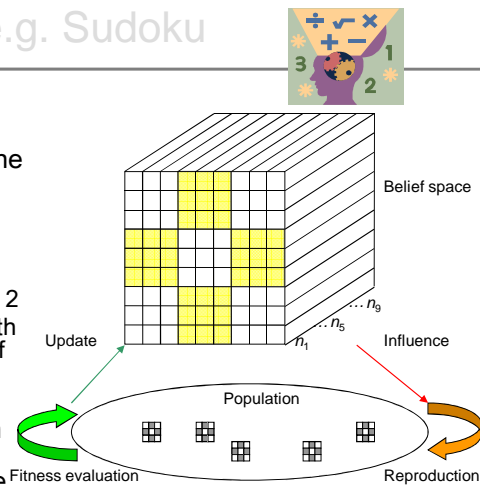


UNIVERSITY of VAASA
Communications and Systems
Engineering Group

CA, belief space e.g. Sudoku

The belief space in this case was a $9 \times 9 \times 9$ cube, where the first two dimensions correspond to the positions of a Sudoku puzzle, and the third dimension represents the nine possible digits for each location

- After each generation, the belief space is updated if:
 - 1) The fitness value of best individual is 2
 - 2) The best individual is not identical with the individual that updated the belief space previous time
- The belief space is updated so that the value of the digit that appears in the best Sudoku solution is incremented by 1 in the belief space.
 - This model also means that the belief space is updated only with near-optimal solutions (2 positions wrong)
 - This information is used only in the population reinitialization process



When population is reinitialized, positions that have only one non-zero digit value in the belief space are considered as givens, these include the real givens and also so called "hidden" givens that the belief space have learned, i.e. those positions that always contain the same digit in the near-optimal solutions



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Cultural algorithms

- If you are interested to know more about cultural algorithms, see slides from Reynolds
 - http://complex.wayne.edu/Seminars_Fall_05/culturalalgorithm_1108.ppt
- Wikipedia
 - http://en.wikipedia.org/wiki/Cultural_algorithms
- Lamarckism
 - <http://en.wikipedia.org/wiki/Lamarckism>
 - <http://en.wikipedia.org/wiki/Meme>
 - http://en.wikipedia.org/wiki/Genetic_memory



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Cellular automata and cellular EAs

- This example discusses of the possibilities of using the Cellular Automata (CA) and the Cellular Genetic Algorithm (CGA) for the object and pose recognition problems, and for the image classification problem.
- The CGA is a Genetic Algorithm (GA) that has some similarities to cellular automata. In CA the cell values are updated with the help of certain rules that define the new value according the current value of the cell and the values of its neighboring cell.
- The CGA has similar cellular structure as CA, but different cell value update method. In CGA the cell value is updated according to the local fitness function, also all cells in the CGA can have different fitness functions.
- The cell update is done by a kind of one generational GA, where GA population contains the individual (or value) of current cell, its neighboring cells and preliminarily defined number of values that are generated from the current cell and its neighbors by crossovers and mutations. The neighborhood can be *e.g.* 3×3 or 5×5 cells and the total local population size correspondingly *e.g.* 16 or 40 items.
- http://en.wikipedia.org/wiki/Cellular_automata
- http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Cellular automata and cellular EAs

- The CGA is tested for object and pose recognition problems so that we have a database of comparison objects, which are usually called the training set. With this method they are rather a comparison set or an example set, since we do not train the method beforehand.
- In this method we use raw gray pixel data for comparison. Thus, in this case the local fitness function for each cell is simply the difference of the pixel values in the corresponding locations of the tested object and the example object.
 - Note that this fitness function only determines the cell update, not the comparison result of the objects. The morphing is done synchronously, so that only the cell values after the previous update round have effect on the cell in this update round, *i.e.* all the cells are updated simultaneously.
- The unknown object is defined to be the closest to object towards which the CGA morphing needs *e.g.* the shortest time, the lowest number of cell updates, or the lowest number of update rounds.
- Unlike normal CA the cell update is not directly dependent on neighbouring cell values. Instead the cell is updated with the neighbouring cell value that get the best evaluation value according the local fitness function.



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

CA and CGA based morphing 11.3.2015

The Problem:

Ordering images into the meaningful order

The Proposed Solution:

Morph images with cellular automata or cellular genetic algorithm and use the amount of transforms they need to morph from image to image as a image distance measure see earth mover's distance

http://en.wikipedia.org/wiki/Earth_mover%27s_distance

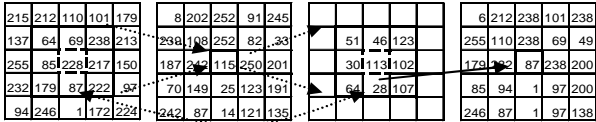


Fig. 1. The cellular automata (CA) version of the proposed method. We calculate the differences of the local neighborhood cells against the target value and the closest value will be the new value of the current cell

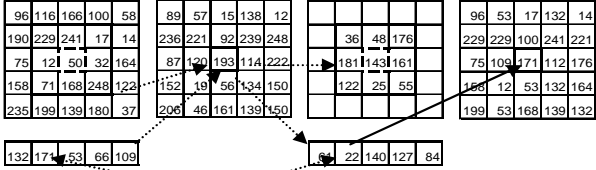




Fig. 2. The cellular genetic algorithm (CGA) version of the proposed method. The current cell under update will get the closest value from either from {local neighborhood cells against the target value} as in CA, or from {genetically generated values against the target value}, as seen at the bottom




UNIVERSITY of VAASA
Communications and Systems
Engineering Group


Morphing examples with CA and CGA




a) From



b) To




c) Result




d) Difference |b-c|


Cellular automata



a)




b)



c)

d)

Cellular genetic algorithm



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Co-evolution

- Co-evolution can be
- Predator-pray type
 - Evolutionary arms race
(http://en.wikipedia.org/wiki/Evolutionary_arms_race)
 - E.g. when the pray evolves faster, also the predator must faster to catch the pray
 - Forms upward spiral where both species evolves because of the other
- Host and a parasite type
- Host and a symbiont type
 - E.g. bees and the pollination of flowers



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Co-evolution


- Co-evolutionary generally means that an evolutionary algorithm is composed of several species with different types of individuals.
- Other organisms are among the most important parts of organism's environment. Co-evolution occurs when two species adapts to their environment, evolve, together. The goal is to accomplish an upward spiral, an arms race, where both species would achieve ever better results.
- The co-evolution is mostly applied to game-playing and artificial life simulations.
- Our examples shows implementation of co-evolution to the simultaneous test image and image filter development and testing, and also surface measurement and test surface development
 - Both are quite special applications, so therefore these approaches are mostly just a proposals at this point.



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

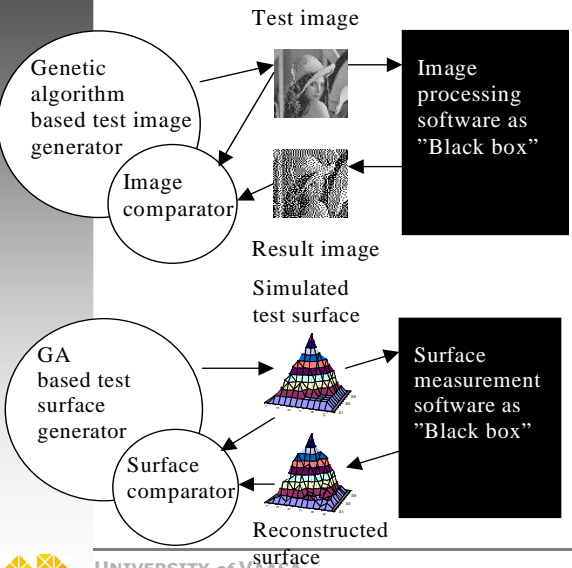
Co-evolution approach, motivation

- The idea of applying co-evolutionary methods for these examples was motivated by previous experience with
 - 1) generating halftoning filters with GA
 - 2) generating test images for testing halftoning methods
 - 3) *generating software test data with GA*
 - 4) *the goal of achieving better software*
- The goals 1 – 2 and also 3 – 4 seem like natural co-evolutionary pairs where the simultaneous optimization against the opposite goal could lead to the co-development




UNIVERSITY of VAASA
Communications and Systems
Engineering Group

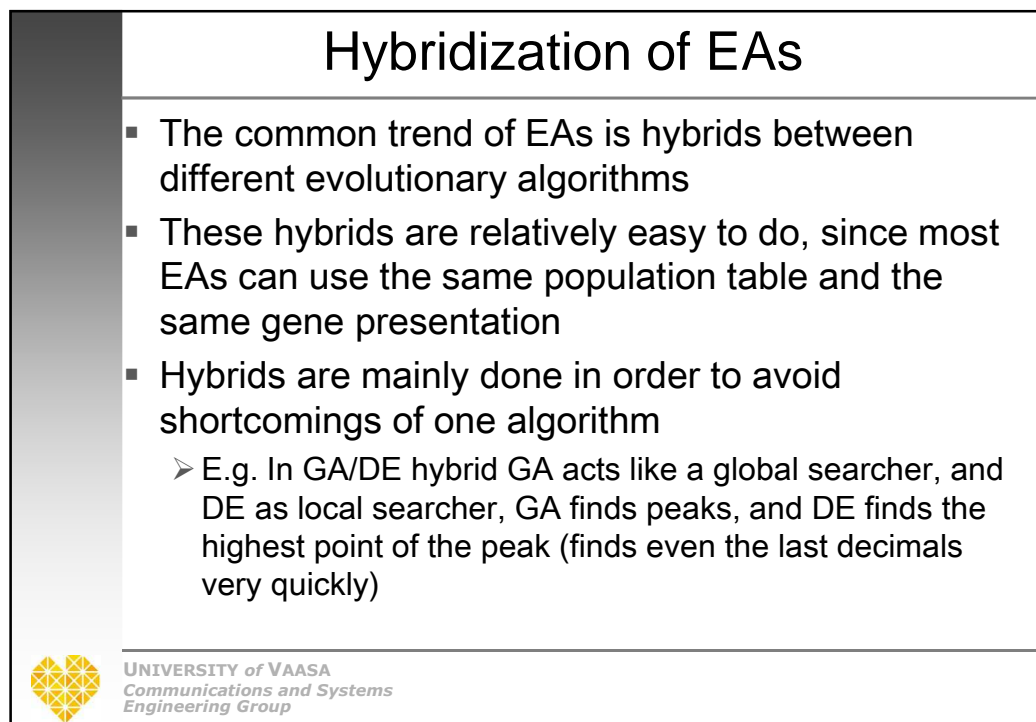
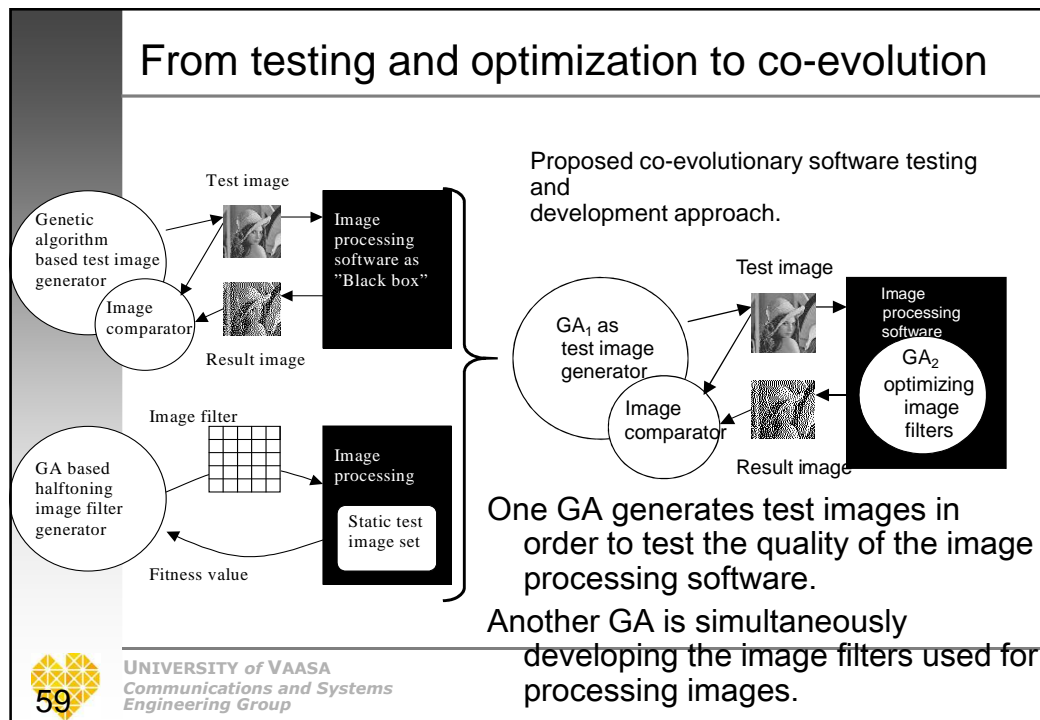
Examples – from optimization to co-evolution



- Image processing software quality is tested by generating test images by GA and measuring the difference between the original and the result image
- Accuracy of machine vision based measurement software is tested by generating simulated test surfaces by GA and calculating the measurement error



UNIVERSITY of VAASA
Communications and Systems
Engineering Group



Example of the Hybridization of EAs (DE+GA)

Current population

Next generation

- The likelihood of doing GA type reproduction is e.g. 30%
- The likelihood of doing DE type reproduction is e.g. 70%
- Population size e.g. 100
- DE parameters can be e.g.
 $F=[0.1, 0.7]$
 $CR=[0.0, 0.9]$
- Mutation probability with GA can also change, e.g. decreases if the best result improves, and increases if the result is not improving

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

The flow chart of the possible DEGA hybrid

```
graph TD
    Start([Start and generate the initial population randomly]) --> EvalPop[Evaluate the population]
    EvalPop --> SortPop[Sort the population, first the feasible solutions according to the fitness value f(x), then the infeasible solutions according to the amount of constrain violations Σg(x)]
    SortPop --> StopCond{Stop condition reached?}
    StopCond -- Yes --> PrintBest([Print the "best" solution and stop])
    StopCond -- No --> LoopStart(( ))
    subgraph Loop [do as many as the population size-elitism]
        direction TB
        SelectMethod{Select the crossover method randomly}
        SelectMethod -- GA --> GAProc[Select randomly two parents and do GA type crossover and mutation]
        SelectMethod -- DE --> DEProc[Select randomly four parents and do DE type crossover-mutation operation]
        GAProc --> EvalNew[Evaluate the new individuals, if the child fulfills one of conditions (right) it will replace the parent it is compared with.]
        DEProc --> EvalNew
    end
    EvalNew --> SortPop
    LoopStart --> SelectMethod
```

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Notes

- The finding of good GA and DE parameters and their ratio is very difficult, and usually even the final compromises does not work ell with all test problems
- Our experiments (in VY) have showed that the DEGA method works relatively well with the constrained test problems. It reaches the feasible region fast and consistently, but
 - Problem: The results were relatively good when compared with the other methods with some the benchmark problems, usually the best test runs obtained good results, however, the average and worst results were not as good, so the method was inconsistent
 - Solution? Need to do further analyze of what characteristics of those problems that causes the obstacles for hybrid DEGA and improve the method accordingly



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Multi-objective optimization

- Sometimes we have more than one optimization goals, and they can be conflicting
 - We are trying to find a compromise that fulfils all different objectives as satisfactorily as possible
 - This kind of problems are e.g.
 - ☞ Minimizing weight while maximizing the strength of some structure in mechanics and building engineering
 - ☞ Maximizing performance while minimizing fuel consumption in cars
 - ☞ Maximizing profits while minimizing risks in economics etc.
 - ☞ The final solution is clearly a compromise between conflicting goals



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Pareto optimization

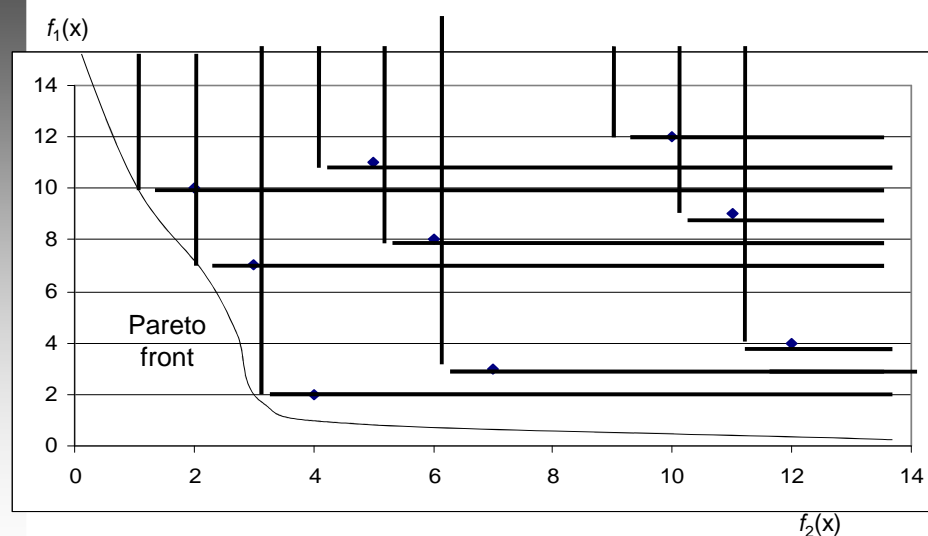
- In Pareto-optimization all points that are dominated by some other point are not 'Pareto-optimal'
- Only the points that are not dominated by any other point are 'Pareto-optimal' and they form 'Pareto-front'
- In the next slide we have two objectives, and both are minimized. Therefore we can draw lines and all the points inside the lines are dominated by the point where the lines begin
 - http://en.wikipedia.org/wiki/Multi-objective_optimization
 - http://en.wikipedia.org/wiki/Pareto_efficiency



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

11.3.2015

Pareto front



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Notes!

- It is important to give different algorithms equal change, when their results are compared
- This applies to all algorithms we might compare (from Eiben and Smith slides 'Working with EA's')
- Equal change means:
 - Usually:

$$\text{time}_{\text{algorithm1}} = \text{time}_{\text{algorithm2}}$$
 - Or sometimes:

$$\text{tested_trials}_{\text{algorithm1}} = \text{tested_trials}_{\text{algorithm2}}$$
 - Or sometimes (rarely, requires the same or justified population sizes):

$$\text{generations}_{\text{algorithm1}} = \text{generations}_{\text{algorithm2}}$$



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Warnings and suggestions

- The benchmark set is important when testing different algorithms
- When you compare your results with others you should use the same benchmark problems as other people
- The test problem choice has a big influence on the results.
 - This was clearly demonstrated by Morovic and Wang. They showed that by selecting an appropriate five-image subset from a 15-image test set, any one of the six Gamut Mapping Algorithms tested could appear like the best one
 - They studied 21 research papers on GMAs and found out that these papers have represented results by using only 1 to 7 test images
- So, NEVER use the limited test set (or subset of the whole benchmark set)



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Parameter control

- Evolutionary algorithms tend to have several control parameters
 - Population size
 - Amount of elitism
 - Mutation rate
 - Crossover rate (portion of different crossover types)
 - Etc.
- With different problems the different parameter setting work best
 - It is difficult to find the combination of parameters that work best for the problem in hand



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Parameter control

- There are different ways of setting the parameters
 - Good guess
 - ☞ choose parameters that seems reasonable according the previous experience
 - Preliminary testing
 - ☞ Test different combination of parameters with shorter optimization runs and choose the most promising
 - Self-adaptive
 - ☞ The parameters are coded into chromosome, so that EA optimizes them together with the problem
 - Meta-GA approach
 - ☞ Run preliminary tests so that upper-level GA optimizes the parameters of secondary GA
 - Success-rule based and time-based parameter controls
 - ☞ See Eiben&Smith slides



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Preliminary testing

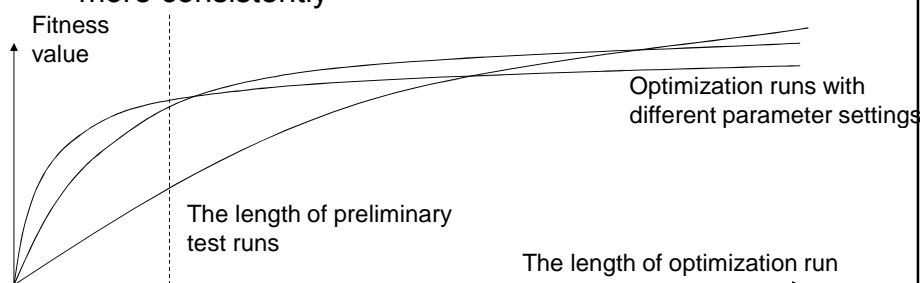
- If we want to test combination of settings, *e.g.*
 - Population size: {10, 20, 30, 40, 50, 60, 80, 100, 120, 200}
 - Elitism: {5, 10, 20, 30, 40, 50} %
 - Mutation probability: {0.01, 0.02, 0.03, 0.05, 0.1}
 - One-point/uniform crossover ratio: {0, 0.25, 0.5, 0.75, 1.0}
- Testing all possible combinations of parameters would require: $n_{\text{psize}} * n_{\text{elit}} * n_{\text{mut}} * n_{\text{cross}}$ preliminary test runs
 - so in the upper example: $10 * 6 * 5 * 5 = 1500$ test runs
- Because the control parameters have mutual influence, usually all different combinations must be tested.
 - Often leads too heavy preliminary testing and we must run preliminary tests short and with quite limited sets of parameter values



UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Preliminary testing

- One problem with preliminary testing is that we usually do them with shorter optimization runs than actual optimization
 - The same parameters may not work with the actual longer run, because some parameter setting are more greedy -> optimization progress aggressively in the beginning, then stop evolving, other parameter setting evolve slowly but more consistently



UNIVERSITY of VAASA
Communications and Systems
Engineering Group