Vaasan yliopisto
UNIVERSITY OF VAASA

Telecommunications seminar

"Evolutionary algorithms in Communications and systems"

Timo Mantere

Professor
Communications and systems engineering

University of Vaasa

3.3.2015

Telecommunications seminar

- TELECOMMUNICATIONS SEMINAR
- Tietoliikennetekniikan seminaari
- Code: TLTE3090
- Credits: 3–10 ECTS (3–10 op)
- Prerequisites: related subject studies on telecommunication engineering
- Learning Outcomes: the aim of this course is to introduce research oriented topics in telecommunications and systems, after completing this course successfully, the student will be able to seek scientific information and to prepare and give seminar presentations, moreover, they will be able to demonstrate the principles of the seminar topic
- Content: this course has varying contents, the current content is always indicated by the course subtitle presented in the course website
- Study Materials: 1. depend on the topic
- Teaching Methods: depending on the topic
- Modes of Study: attending seminar sessions, quizzes, preparing scientific report and giving at least one presentation
- Languages: english
- Grading: 1–5 or failed, or passed/failed (depends on the topic)
- Responsible Person: Mohammed Elmusrati and Reino Virrankoski
- Teacher(s): Mohammed Elmusrati, Reino Virrankoski
- Responsible Unit: Department of Computer Science
- Additional Information: annual course, website cs.uwasa.fi/courses/tlte3090

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

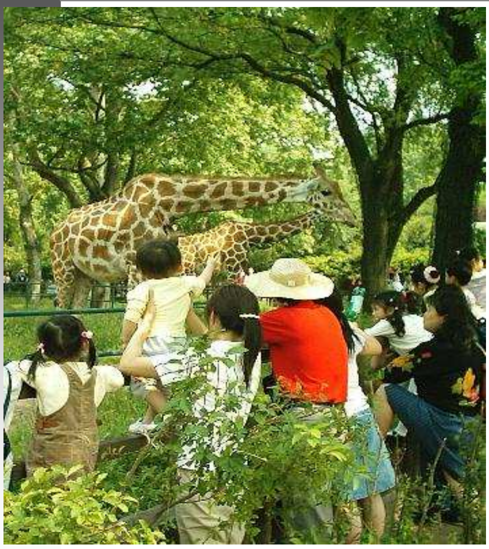
Evolutionary algorithms in communications

INTRODUCTION TO EAs

- What are evolutionary algorithms?
- Nature vs. Eas
- History
- Problem coding
- Fitness landscape
- No free lunch

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Nature



- In nature all the plants and animals are constructed by the instructions from the common basic elements – the genes
- Combinations of these basic structures can generate a huge variety of different kind of organisms
- *E.g.* humans and chimpanzees differs genetically by 0.7- 4.8% depending the how the difference is determined (http://en.wikipedia.org/wiki/Chimpanzee).
- See more information about genetics from: http://en.wikipedia.org/wiki/Genetics

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Background

- Evolutionary computing (EA) is a field of science and engineering that tries to apply some phenomena's that appears in the nature to the optimization
- Most notable is the adaptation of Charles Darwin's evolution theory in order to solve difficult search and optimization tasks
  - Method is universally applicable, since it have been successfully applied to almost all thinkable search and optimization problems in engineering, science and people's everyday life.
  - This course will present the basic working principles of evolutionary algorithms and what things have an effect to the algorithm's efficiency.
  - We will also present some applications of EAs.
  - Genetic algorithms, genetic programming, evolution strategies, evolutionary programming, and partly differential evolution also are based on this theory

Background

- Other notable EAs are based on herd behaviour of animals
  - Ant colony optimization (ACO) adapts some principles of ant behaviour, pheromone paths etc
  - Particle swarm optimization adapts some principles of bird and fish swarms and how they "follow the leader"
- Cultural algorithms and memetic algorithms tries to mix some cultural and learned knowledge into genetics, usually these are composed of basic GA with added cultural components
- This course will offer a technical view (especially the computer science and automation view) to the biology, genetics and the evolution
- More information about Darwin and the evolution:
  - http://en.wikipedia.org/wiki/Charles_Darwin
  - http://en.wikipedia.org/wiki/Introduction_to_evolution
  - http://en.wikipedia.org/wiki/Evolution_theory

Evolutionary algorithms

- Evolutionary algorithms are part of computer science and ICT (Information and Communications Technology)
- Evolutionary algorithms are not part of genetics, biology or biosciences
- Evolutionary algorithms have however got their inspiration from the biological processes and terminology
  - Population, individual, parents, offspring, generations, gene, chromosome, crossover, mutation, ”survival of the fittest” etc.
- EAs can be applied to bioscience research
- The main weakness of the EA methods are that there is no complete theory that would explain their operation and efficiency

Evolutionary algorithms

Evolution		Problem solving
Environment		Problem
Individual		Solution candidate (trial)
Fitness		Quality
Fitness	->	will define the probability to survive and reproduce
Quality	->	will define the probability to act as a model/basics to the new solution candidates

- Individuals are points in the search or fitness space, together the population individuals represent a cloud of points that moves around the problem landscape (search space), while they evolve and adapt

History

- 1948, Turing propose genetic or evolutionary search
- 1962, Bremermann did some optimizing with the help of evolution and crossover
- 1964 Rechenberg presented ”Evolution strategies”
- 1965, Fogel, Owens and Walsh presented ”Evolutionary programming”
- 1975, Holland presented ”Genetic Algorithms”
  - Finally causing the big boom in research, and the new ever-growing research area of EAs was really born
- 1992 Genetic programming (Koza),
- 1992 Ant colony optimization (Dorigo),
- 1994 Cultural algorithms (Reynolds),
- 1995 Particle swarm optimization (Kennedy&Eberhart),
- 1996 Differential evolution (Storm&Price) etc.

Nature

- Different organisms establish the equilibrium of the nature
  - If there’s an ecological niche available in the ecosystem, some species will soon evolve and occupy that open niche
  - http://en.wikipedia.org/wiki/Ecosystem
  - http://en.wikipedia.org/wiki/Ecological_niche
- DNA have a lot of so-called ”junk-dna”, that does not code proteins, therefore it is difficult to say exactly how far or close some species DNAs are from each other
  - http://en.wikipedia.org/wiki/Junk_DNA
- Interesting book about evolution:
  - The Making of the Fittest: DNA and the Ultimate Forensic Record of Evolution by Sean B. Carroll
  - http://www.amazon.com/The-Making-Fittest-Ultimate-Evolution/dp/0393330516

Nature

- In the nature the genotype (genetic information) and phenotype (the properties of an individual) does not decode directly, but also e.g. the environment does affect to the phenotype
    - Less food, less energy, less genetically inherited options fully develops
    - Accidents, illnesses etc. Affect on individual
  - Several genes affect to one property of an individual (*polygene*).
  - One gene affects several properties (*pleiotropy*).
    - http://en.wikipedia.org/wiki/Genotype
    - http://en.wikipedia.org/wiki/Phenotype
    - http://en.wikipedia.org/wiki/Polygene
    - http://en.wikipedia.org/wiki/Pleiotropy

Problem coding

- An important part of evolutionary optimization is problem coding
- Coding must fit to the problem, eg.
  - Binary coding if we are using binaries or integers
  - Floating point coding, if we are handling real numbers
  - Often problem presented as matrix or N:th dimensional array
  - Special codings if we are searching for order etc.

- In all cases coding must be able to present all possible solutions
- Coding must allow traveling from each point in search space to any other point in search space
- Usually coding should not be able to present impossible situations, however, this is often unavoidable, and need to be handled with penalty functions

Genotype to phenotype

- In the nature genotype and environment together forms the properties of an individual (phenotype)
- In computer science same bit sequence (genotype) can be interpret to the different properties (phenotypes)
  - E.g. bit vector 001101010 (genotype) can represent (phenotype)
    - Bit vector: 001101010
    - hexadecimal: 6A (0 0110 1010)
    - Octal decimal: 152 (001 101 010)
    - Integer: 106 $(=1*2^6+1*2^5+1*2^3+1*2^1)$
    - Real number: 6.625 $(0110.1010 = 1*2^2+1*2^1+1*2^{-1}+1*2^{-3})$
    - ASCII character: j (106=6a= j)
    - Something else, *e.g.* route: Espoo->Oulu->Tampere (001 101 010=152, where {0=Helsinki, 1=Espoo, 2=Tampere, 3=Vantaa, 4=Turku, 5=Oulu, 6=Lahti, 7=Kuopio}

Optimization problem

- Usually we have a function which properties are unknown
  - The optimum (maxima or minima) is unknown
  - The fitness landscape is unknown
  - Linearity, continuity, unimodality, separability, etc. are unknown
- So we have a global non-linear optimization problem to solve

What makes a function difficult?

EASIER:	MORE DIFFICULT:
Linear	Nonlinear
Separable	Inseparable
Unimodal	Multimodal
Serializable	Non-serializable
Unconstrained	Constrained
Continuous	Discontinuous
Few parameters	Many parameters
Small search space	Large search space

Prerequisites for using EAs

- There are some Prerequisites for applying evolutionary algorithms for a certain problem:
  - Problem is not linear, separable or serializable
  - Large solution space
  - Not known efficient and complete solution
  - Sufficient level of continuity
  - Existence of suitable fitness function candidates
  - Cheap testing of candidate solutions

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Prerequisites for using EAs

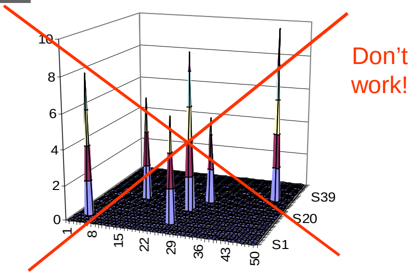
- Problem is not linear, separable or selializable
  - Linear problems are easily solved with other methods faster and more accurately
  - In separable problems, it is more efficient to solve parameters one by one, and basic GA operation, crossover, becomes useless
  - Serializable problem is also solved more efficiently by searching parameter values one by one
    - There may be some special cases where separable or serializable problem is still difficult enough for other methods and GA may become useful
- Large solution space
  - The solution space must so large that it cannot be enumerated through with reasonable time or expenses
- No known efficient and complete solution
  - If the problem is already solved, there is not much to gain
  - If only partial solutions are known, EAs may help to ”fill the gaps”

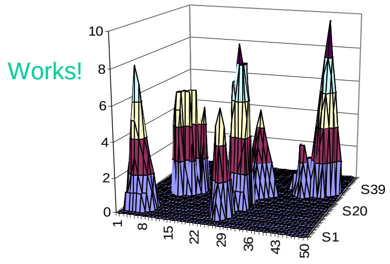
Prerequisites for using EAs

- Existence of suitable fitness function candidates
  - A meaningful fitness metric that can lead the search towards better solutions according to that metric
  - If many different metrics can be used, the multi-objective approach is possible
- Cheap testing of candidate solutions
  - EA methods typically generate a large number of candidate solutions, therefore the solution candidates should be evaluated in reasonable time with reasonable expenses

UNIVERSITY of VAASA
Communications and Systems
Engineering Group

Fitness landscape





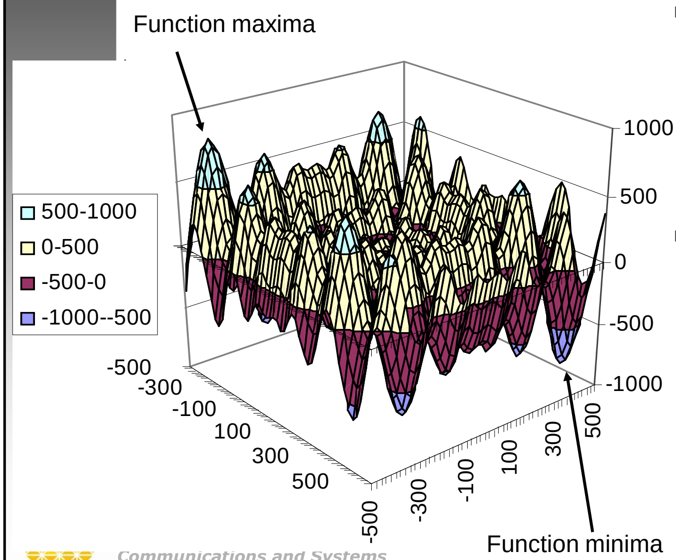
If searching system problems that does not have any "symptoms" nearby optimum, "Needle in the haystack" problem

GA cannot adapt to the environment

Landscape that does indicate "symptoms" around the local optimums

GA can adapt to the environment and gain advantage over random methods

The search space



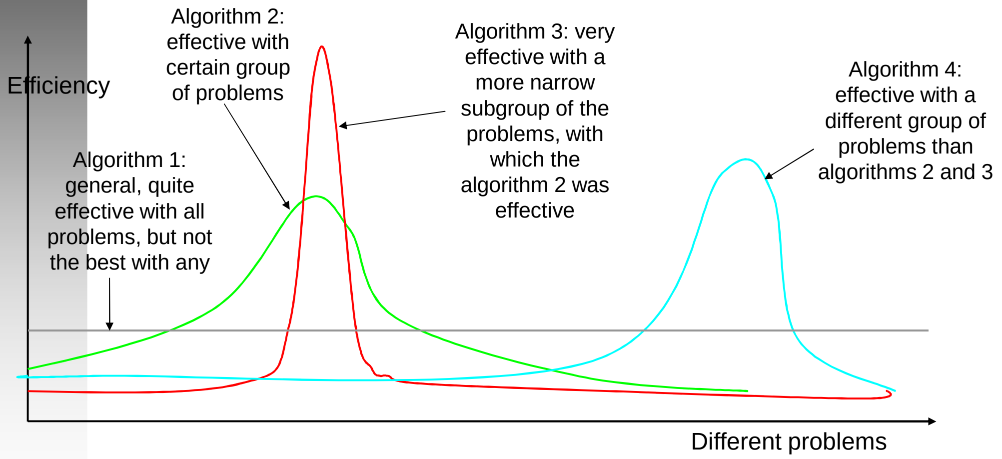
- Usually the search space is multi-modal and we do not know in which peak the global optima is
- Usually we optimize only one goal, the minimum or maximum of a function
  - There exists EAs that tries to find both simulta-neously

No Free Lunch

- Wolpert and Macready have presented so called ”no free lunch” theorem, it claims that all algorithms are as good, if compared with all possible problems
- I practice this means that some algorithms are better for some group of problems and other with other problems.
  - Some algorithms are tailored (with problem specific knowledge) to the limited set of problems with which they are very efficient, but then they are also quite worthless with other problems
  - All the algorithms may have the group of problems, with which they are the most effective algorithm
  - General optimization methods may be effective for a large group of problems, but maybe they are not the most effective algorithm for any problems

No free lunch

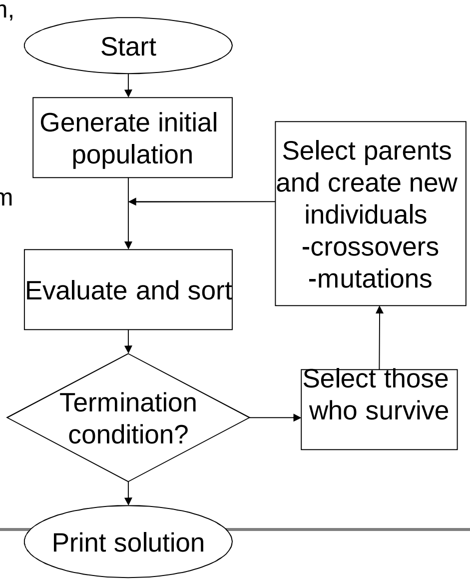
- Different algorithms are effective with different problem groups



Genetic Algorithm (GA)

The flow chart of a typical genetic algorithm, but applies to many other Eas too

1. Create initial population
2. Evaluate population (sometimes also Sort the population)
3. Test the termination condition (optimum or good enough solution found, or enough iterations done)
4. If not stop, select those who survive (elitism)
5. Generate new individuals with crossovers and mutation
6. Continue from step 2



UNIVERSITY of VAASA
Communications and Systems Engineering Group

Initial population

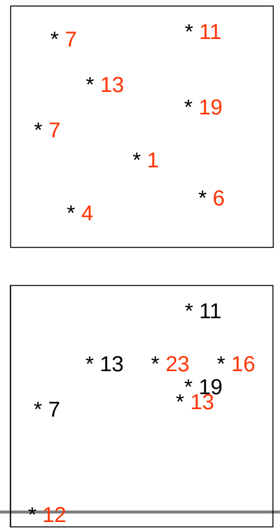
- Usually at the beginning we want to fill the solution space as widely and completely as possible
- The initial population is usually generated randomly
- It is also possible to seed some good known solutions to the initial population
- Sometimes we search the problem with some other (faster, but less accurate) method, in order to find relatively good solutions and the seed those to the initial population, and try to evolve them better with EA

UNIVERSITY of VAASA
Communications and Systems Engineering Group

Hit and miss

- Evolutionary algorithms are population based ”hit-and-miss” algorithms
- First we kind of ”shoot with the shotgun” points to the whole target (search space)
- Then we come closer and aim towards the best value
  - Like with shotgun most of will now go close to the spot we aim, but some led balls can be spread throughout the whole target

First shot



Second shot, old individuals as black, and new as red

UNIVERSITY of VAASA
Communications and Systems Engineering Group

Crossover and mutation

- The crossover is usually done so that we favour those individuals that have the highest fitness values, i.e. they are more likely to be selected as parents
- Crossover can be performed e.g. as one-point crossover, where we take the beginning from the first parent and the end the second parent
- In mutation we change some randomly selected gene randomly (but so that it still stays within the boundaries of the parameter, e.g. [A, z] or [0, 1] etc.

Crossover:

a	b	c	d	e	f	g	h	I	j
k	l	M	n	o	p	q	r	S	t

↓

0	0	0	0	0	1	1	1	1	1

↓

a	b	c	d	e	p	q	r	S	t
k	l	M	n	o	f	g	h	I	j

Mutation:

a	b	C	d	e	f	g	h	I	J

↕ Mutaatiokohta

a	b	C	d	e	v	g	h	i	J

UNIVERSITY of VAASA
Communications and Systems Engineering Group

Binary coded GA

- In binary coded GA the individuals chromosome is consisted of bitvector
- In uniform crossover with each bit we randomly select from which parent the bitvalue is taken
  - We can create either one or two new individuals (in the case of 2 the other one possess the opposite parental genes as the first one)
- In binary coded GA the mutation means flipping the bit value (either 0->1 or 1-> 0)

Crossover:

```
1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1     Parents

1 0 0 1 1 0 1 0 1 1     Crossover vector (we can be also draw and use random values immediately)

0 1 1 0 0 0 1 0 1 1
1 0 0 1 1 1 0 1 0 0     Offspring = new individuals
```

Mutations:

```
0 1 1 0 0 0 1 0 1 1     The individual to be mutated

0 1 0 0 0 1 1 0 1 1     The mutated new individual
```


Floating point coded GA

Parent 1

1.51	*5.77*	*3.12*	*0.12*	*3.00*	*5.51*

one-point crossover

Mutation

4.33	7.11	9.52	**4.44**	2.00	0.11

Parent 2

⇩

Child 1

1.51	*5.77*	9.52	**6.66**	2.00	0.11

4.33	7.11	*3.12*	*0.12*	*3.00*	*5.51*

Child 2

Mutation and crossover operations in the floating point coded GA

- We have a vector of real numbers
  - in crossover we take each number from one parent
  - In mutation we randomly draw the new value within the boundaries of the values e.g. [0.0, 10.0]

Real coded GA

- In the floating point coded GA the chromosome of GA consist of real numbers that are within some boundaries, e.g. [0, 1]
- In real coded GA we can use one-point, multipoint, uniform or arithmetic crossovers
- In real coded GA the mutation can be random between the boundaries or Gaussian (adding Gaussian distributed random number to the current value)
- It is also possible to use binary coded GA and interpret the bit vector into floating point numbers

Arithmetic crossover and **mutation:**

0.12 0.15 0.72 0.66 0.98 0.11 Parent 1
0.56 0.76 0.28 0.99 0.55 0.88 Parent 2

0.34 0.46 0.50 **0.43** 0.77 0.50 Child 1
0.21 **0.78** 0.63 0.73 0.89 0.26 Child 2

The child 1 formed: crossover by arithmetic mean:

$\frac{Gene_{Parent1} + Gene_{Parent2}}{2}$

The child 2 formed: crossover by weighted arithmetic mean:
0.8*GeneParent1+0.2*GeneParent2

The mutated gene in **bold** and is most likely result of random mutation in child 1 (large change) and Gaussian mutation in child 2 (small change)

UNIVERSITY of VAASA
Communications and Systems Engineering Group

Crossover without mutation?

- The ”crossover only” EAs do not work, because during the crossover some genetic information is lost, and the population will not obtain new information without a mutation
  - Below is the example why crossover only won’t work; if we are optimizing all-ones and there does not exist value 1 for some gene in the current population, the optimum can never be reached without mutation and new possible value for that gene location

Population of 4 individuals The gene values in 8 gene locations:

Individual 1:	1	1	0	1	1	0	0	1
Individual 2:	1	1	0	0	0	1	0	1
Individual 3:	1	0	0	1	1	1	0	1
Individual 4:	1	1	0	1	0	0	0	0
The possible values of each gene location after the crossovers:	1	[0, 1]	0	[0, 1]	[0, 1]	[0, 1]	0	[0, 1]

UNIVERSITY of VAASA
Communications and Systems Engineering Group