OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

# Evolutionary Algorithms

## Applications of evolutionary algorithms

**Prof. Dr. Rudolf Kruse** **Pascal Held**

{kruse,pheld}@iws.cs.uni-magdeburg.de

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik

Institut für Wissens- und Sprachverarbeitung

# Outline

# Parallelization of evolutionary algorithms

EA: fairly **expensive optimization methods** since one often has to work with

- large population (a few thousand up to several tens of thousands of individuals)
- large number of generations (a couple of hundreds)

**Advantage**: slightly better solution quality compared to other approaches

**Disadvantage**: unpleasantly long execution time

one way to improve this: **parallelization**

distribution of necessary operations on several processors

Questions:

- **Which steps can be parallelized?**
- **What additional, specialized techniques are inspired by a parallel organization of the algorithm?**

# Which steps can be parallelized?

**Creating an initial population:**

- often easy to parallelize, because usually the chromosomes of the initial population are created randomly and independently of each other
- attempt to prevent duplicate individuals may pose obstacles to a parallel execution
  fairly little importance overall though, because the initial population is created just once

**Evaluation of chromosomes:**

- easily parallelizable because usually an individual is evaluated independently of any other ones
- even in prisoner's dilemma: process pairings in parallel

**Computing the (relative) fitness values or a ranking of the individuals**

- central agent that collects and processes evaluations

# Which steps can be parallelized?

**Selection:** whether the selection of the individuals is parallelizable, depends heavily on the chosen selection method

- **expected value model** and **elitism:**
  all require to consider the population as a whole and therefore are difficult to parallelize
- **Roulette-wheel** and **rank-based selection:** can be parallelized after the initial step of computing the relative fitness values
- **Tournament selection:** best suited for a parallel execution, especially for small tournament sizes, because all tournaments are independent and thus can be held in parallel

# Which steps can be parallelized?

**Applying genetic operators**
can easily be applied in parallel, since they affect only one (mutation) or two chromosomes (crossover), and are independent of any other chromosomes (If combined with tournament selection, a steady-state evolutionary algorithm can thus be parallelized very well)

**termination criterion:**
simple test whether a certain number of generations is reached does not cause any problems
Termination criteria like

- the best individual of the population exceeds a user-specified fitness threshold, or
- the best individual has not changed (a lot) over a certain number of generations

need a central agency that collects this information about the individuals

# Island Model

- if we require a selection method that causes some troubles for parallelization:
- achieving a parallel execution by simply processing several independent populations
- each population can be seen as inhabiting an island, therefore **island model**.
- pure island $\equiv$ executing the same evolutionary algorithm multiple times
- yields results that are somewhat worse than those of a single run with a larger population

# Migration

one may consider exchanging individuals between the island populations at certain fixed points in time (*not* in every generation)

**Migration** (Wanderung)

usually no direct recombination of chromosomes from different islands

after migration: Recombination of genetic material of one island with another

# Control the migration between islands

**Random model:**

pairs of islands are chosen randomly, which then exchange some of their inhabitants

any two island can, in principle, exchange individuals

**Network model:**

Islands are arranged into a network or graph

Individuals can migrate only between islands that are connected by an edge in the graph

Along which of the edges individuals are exchanged is determined randomly.

# Contest model

- evolutionary algorithms that are applied on the islands differ in approaches and/or parameters
- population size of an island is increased or decreased according to the average fitness of its individuals
- usually: a lower bound for the population size is set

# Cellular Evolutionary Algorithms

**also called: "isolation by distance"**

Processors are arranged in a rectangular grid

usually in the shape of a torus in order to avoid boundary effects

Selection and crossover are restricted to adjacent processors (to those connected by an edge), mutation to single processors

Example: each processor is responsible for one chromosome

- **Selection:** processor chooses the best chromosome of the (four) processors adjacent to it (or one of these chromosomes randomly based on their fitness)
- **Crossover:** processor then performs crossover of the selected chromosome with its own and may also mutate the chromosome (The better child resulting from such a crossover replaces the chromosome of the processor)

construction of groups of adjacent processors that maintain similar chromosomes

mitigates the usually destructive effect of the crossover

# Approach of Mühlenbein

**Combination** of EAs **and hill climbing**

- every individual is optimized with hill climbing, that is
  - random mutations are applied and kept if they are advantageous
  - otherwise they are retracted
- hill climbing can be easily parallelized
- individuals search for a crossover partner in their (local) neighborhood (this requires a distance measure for the individuals — relates to niche techniques, e.g. *power law sharing*)
- offspring perform local hill climbing
- individuals of the next generation are selected with a **local elite principle**

$\Rightarrow$ best two individuals among the four involved individuals (two parents and two children) replace the parents

# Outline

# Random numbers

EAs base in general on random changes of exisiting solution candidates and selection

all created „Random numbers“ $\neq$ really random

created by determinic algorithm: **Pseudorandomness**

- attempts to use random behaviour of physical processes numbers with worse properties than deterministic methods
- on top: reducability of simulations with deterministic methods

# Historical

**aus [Knuth, 1997]**

before 1900: Scientist who needed random numbers drew balls from a „well shuffled urn“, rolled dices or dealed cards

1927, L. Tippet published a table with $\geq$ 40000 random numbers

Disadvantages of tables: expensive to create and store

machines to generate random numbers

mechanic methods: error-prone and no reproducability

arithmet. operations on computer: John von Neumann (1946)

- calculate square of the last random number and extract middle digits
- e.g. 10-place random numbers and last number be 5772156649
- square is 33317792380594909201
- therefore, next number is 7923805949

# Argument against arithmetic methods

Can we call a generated sequence "randomïf every number is completely determined by its predecessor?

*Anyone who considers arithmetic methods of producing random digits is, of course, in a state of sin.* — John von Neumann (1951)

- Answer: sequence is **not** random, but it seems to be!
- deterministic algorithms generate **pseudorandom** sequences
- von Neumanns method has some issues:
  sequence tends to short periods of repeating numbers
- z.B. bei 4-stelligen Zahlen: 6100, 2100, 4100, 8100, 6100, ...
- following: methods which are advantageous in comparison to the von Neumann method

# Generating uniform distributed random numbers

- Creating a sequence of real numbers (uniform-distributed from 0 to 1)
- fixed (floating point) precision actually leads to generating integers $X_n$ between 0 and $m$
- fraction $U_n = X_n/m$ lies between 0 and 1
- in general: $m$ is word size of the computer $w$

# The linear-congruential method

- most popular random number generators are special implementations of [Lehmer, 1951]
- we choose 4 magic integers

| | | |
|---|---|---|
| $m$, | Modules; | $0 < m$, |
| $a$, | Factor; | $0 \leq a < m$, |
| $c$, | Increment; | $0 \leq c < m$, |
| $X_0$, | Initial value; | $0 \leq X_0 < m$ |

- favored sequence of random numbers $\langle X_n \rangle$ by

$$X_{n+1} = (a \cdot X_n + c) \bmod m, \qquad n \geq 0$$

- Remainder mod $m$: localisation of a ball on a rotating roulette wheel

# The linear-congruential method

- for e.g. $m = 10$ and $X_0 = a = c = 7$ yields to

$$\langle X_n \rangle = 7, 6, 9, 0, 7, 6, 9, 0,$$

- so, sequence is not always random for all values of $m, a, c, X_0$
- such loops: on all sequences with $X_{n+1} = f(X_n)$
- **periodic** behaviour
- Objective: useful sequences with a fairly long period

# Favourable choice of parameters

- arbitrary initial value $X_0$
- Module $m \geq 2^{30}$ or greatest prime number smaller $w = 2^e$ whereby $e = \#$ representable bits

$\Rightarrow$ *sequence* with maximum length

- if $m$ Power of 2: factor $a = 2^k + 1$ whereas $2 \leq k \leq w$

$\Rightarrow$ *period* with maximum length

- Increment $c$ is subsidiary if $a$ is well-chosen: but $c$ may not have a common divisor with $m$ (e.g. $c = 1$ or $c = a$)

$\Rightarrow$ avoid multiplication by shift and add operations:

$$X_{n+1} = ((2^k + 1) \cdot X_n + 1) \bmod 2^e$$

- generate at most $m/1000$ numbers

# Further methods

- square method of R. R. Coveyou: let

$$X_0 \bmod 4 = 2, \qquad X_{n+1} = X_n(X_n + 1) \bmod 2^e, \qquad n \geq 0$$

  be
- computable with similar efficiency like linear-congruential method
- Mitchell and Moore (1958) proposed the approach

$$X_n = (X_{n-24} + X_{n-55}) \bmod m, \qquad n \geq 55$$

  whereas $m$ is even and $X_0, \ldots, X_{54}$ is chosen arbitrarly
- very efficient to implement using a cyclic list
- period of $2^{55} - 1 \Rightarrow$ probably best algorithm

# Generating normal-distributed random numbers

Polar method [Box and Muller, 1958]:
let $U_1, U_2$ be independent and uniform distributed from $[0, 1]$
the following random numbers are generated from the same distribution $N(0, 1)$

$$X_1 = \sqrt{-2 \ln U_1} \cos 2\pi U_2, \qquad X_2 = \sqrt{-2 \ln U_1} \sin 2\pi U_2$$

- Proof: inverse relations are

$$U_1 = e^{\frac{-(X_1^2 + X_2^2)}{2}}, \qquad U_2 = -\frac{1}{2\pi} \arctan \frac{X_2}{X_1}$$

$\Rightarrow$ multivariate density of $X_1, X_2$ is

$$\begin{aligned} f(X_1, X_2) &= \frac{1}{2\pi} e^{\frac{-(X_1^2 + X_2^2)}{2}} \\ &= \frac{1}{\sqrt{2\pi}} e^{\frac{-X_1^2}{2}} \cdot \frac{1}{\sqrt{2\pi}} e^{\frac{-X_2^2}{2}} = f(X_1) \cdot f(X_2) \end{aligned}$$

# Generating normal-distributed random numbers

**Algorithm** Polar-method

**Output:** two independent, normal-distributed random numbers $X_1, X_2$

1: **do** $\{$
2: $\quad U_1, U_2 \leftarrow$ generate two independent random numbers from $U([0,1])$
3: $\quad V_1 \leftarrow 2U_1 - 1$
4: $\quad V_2 \leftarrow 2U_2 - 1$
5: $\quad S \leftarrow V_1^2 + V_2^2$
6: $\}$ **while** $S < 1.0$ **and** $S \neq 0$
7: $X_1 \leftarrow V_1 \sqrt{\frac{-2 \ln S}{S}}$
8: $X_2 \leftarrow V_2 \sqrt{\frac{-2 \ln S}{S}}$
9: **return** $X_1, X_2$

# Random numbers for EA

- reasonable for an object-oriented approach:
  one random number generator per individual
- disadvantage: consequences due to randomness cannot be estimated

$\Rightarrow$ pro optimization method: only one random number generator

- a clearly defined initial value is advisable $\Rightarrow$ experiments are reproducable
- system time or last created number as seed should be avoided

# Additional Literature I


Box, G. E. P. and Muller, M. E. (1958).
A note on the generation of random normal deviates.
*Annals of Mathematical Statistics*, 29(2):610–611.

Knuth, D. E. (1997).
The art of computer programming, volume 2: Seminumerical algorithms.
pages 1–193. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 3rd edition.

Lehmer, D. H. (1951).
Mathematical methods in large-scale computing units.
In *Proc. 2nd Symp. on Large-Scale Digital Calculating Machinery*, pages 141–146, Cambridge, MA, USA. Harvard University Press.