Evolutionary Algorithms
Introduction - Organizational

• Prof. dr. Thomas Bäck
  – Head of the Natural Computing Group
  – T.H.W.Baeck@liacs.leidenuniv.nl
  – Room 169
  – Cell: +49 177 329 5153

• Dr. Michael Emmerich - Assistant Professor
  – m.t.m.emmerich@liacs.leidenuniv.nl
  – Room 161
  – Cell: +31-(0)71-5277094

• Zhiwei Yang
  – z.yang@liacs.leidenuniv.nl
  – Room 160

• Kaifeng Yang
  – k.yang@liacs.leidenuniv.nl
  – Room 160

• Web page: http://natcomp.liacs.nl/EA
Introduction - Organizational

• Exam (60%)
• Two Practical Assignments (40%)

• You can work in teams of 2!

• Results written up as (scientific) reports
  – Special Template
  – Exercise writing of scientific articles
Introduction - Organizational

• Lectures and werkcollege!
  – Werkcollege by Zhiwei Yang and Kaifeng Yang
  – Thursday, 13.45 – 15.30
  – Starts Sep.9 (next week)
  – Werkcollege consists of:
    • Workgroup sessions: Exercises in preparation for exam
    • LAB sessions: Exercises as support for Practical Assignments
Introduction – Important dates

• PA1 Deadline: Friday October 23, 5 pm
• PA2 Deadline: Friday November 27, 5 pm
  – Every week late: 0.5 pts grade degradation

• Exam: TBA (probably week of December 10th)
Overview of the Course

- Sep. 3: Introduction
- Sep. 10: Optimization, Evolutionary Algorithms – General Concepts
- Sep. 17: Genetic Algorithms – Basics
- Sep. 24: Genetic Algorithms – Theory (Schema Processing)
- Oct. 1: No lecture
- Oct. 8: Genetic Algorithms – Convergence Velocity (Mutation Analysis),
- Oct. 15: Genetic Algorithms – Applications
- **Oct. 24:** Deadline Practical Assignment 1
- Oct. 29: Evolution Strategies – Self-Adaptation,
- Nov. 5: Evolution Strategies – Theory,
- Nov. 12: Evolution Strategies – Applications
- Nov. 19: Biological Background
- Nov. 26: Genetic Programming
  - **Nov. 28:** Deadline Practical Assignment 2
- Dec. 3: Advanced Topics
Introduction:

Searching big search spaces
Optimization Creating Innovation

• Illustrative Example: Optimize Efficiency
  – Initial:

  – Evolution:

• 32% Improvement in Efficiency!
Background I

Biology = Engineering (Daniel Dennett)
A Simple Task

• Assume a function

\[ f : \{0,1\}^n \rightarrow \mathbb{R}, \quad f(x) = \sum_{i=1}^{n} x_i \]

• Find a vector \( \vec{x}^* \) such that \( f(\vec{x}^*) \rightarrow \max \)

• For \( n = 1000 \ldots \)

• Easy, isn’t it?
A More Complex Task

- Search space: \( \{0,1\}^n \)
- Goal: Find \( \bar{x}^* \in \{0,1\}^n \) such that

\[
E(\bar{x}) = \left( \sum_{k=1}^{n-1} \left( \sum_{i=1}^{n-k} y_i \cdot y_{i+k} \right)^2 \right) \rightarrow \text{min}
\]

\[
y_i = 2x_i - 1
\]

- For \( n = 1000 \)?
Background

- DNA molecule
- Carries genetic information
- Human DNA:
  - $3 \cdot 10^9$ Base pairs
  - $4^{3 \cdot 10^9}$ Combinations
Thought Experiment:

- $10^{80}$ Elementary particles in the universe
- $10^{40}$ Time steps since "big bang"
- $10^{120}$ Possible "computations" in the universe…
- $4^{3 \cdot 10^9}$ is faaaaaaar larger!
Attempt I: Monte-Carlo Search

- Search space: \( \{0,1\}^n \)
- \( 2^n \) possible solution candidates
- \( \bar{a}^* \in \{0,1\}^n \) is the goal vector (say, \((1001100010101)\))

1. \( k := 1; \)
2. *Randomly generate a vector* \( \bar{a}_k \in \{0,1\}^n \)
3. If \( \bar{a}_k \neq \bar{a}^* \) then
   - \( k := k+1; \)
   - goto 2;
fi
Analysis: Monte-Carlo Search

- Prob. to generate $\bar{a}^*$ within a single random experiment:

$$p_1 = P\{\bar{a}_1 = \bar{a}^*\} = 2^{-n}$$

- Prob. to generate $\bar{a}^*$ within $k$ iterations:

$$p_k = 1 - (1 - 2^{-n})^k$$

- Solving for $k$, the number of iterations:

$$k = \frac{\ln(1 - p_k)}{\ln(1 - 2^{-n})} \approx -2^n \ln(1 - p_k)$$

- Here we used $\ln(1 + x) \approx x$ for $x \approx 0$
Result: Monte-Carlo Search

- From the condition \(-2^n \ln(1 - p) \geq 2^n\) we conclude:

- Monte-Carlo search performs worse than complete enumeration for

  \[ p > 1 - 1/e \approx 0.63 \]

- Not a really good idea!
- But: Evolution is often misinterpreted as Monte-Carlo search! [Brooks 1958]
Result: Monte-Carlo Search

• The reason for MC to perform worse than complete enumeration?
• Identical strings might be tested repeatedly
• For genetic information of human beings:
  – \( n = 10^9 \),
  – 4 different symbols,
  – \( 4^{10^9} \) variants
• Conclusion: Evolution is not MC-search!
Attempt II: Evolutionary Search

1. \( t := 1; \)
2. Randomly generate a vector \( \bar{a}_k \in \{0,1\}^n \)
3. Create a copy \( \bar{a}'_k \) of \( \bar{a}_k \)
4. Invert every bit of \( \bar{a}'_k \) with probability \( p \in ]0,1[ \)
5. If ( \( \bar{a}'_k \) matches \( \bar{a}^* \) in more bits than \( \bar{a}_k \) ) then
   \[ \bar{a}_{k+1} := \bar{a}'_k \]
   else
   \[ \bar{a}_{k+1} := \bar{a}_k \]
   fi
6. If \( \bar{a}_{k+1} \neq \bar{a}^* \) then
   \[ k:=k+1; \text{ goto 3; } \]
   fi

Mutation rate (pre-defined)
Analysis: Evolutionary Search

- Assume exactly $m$ bits are still wrong
- Prob. to preserve all $n-m$ correct bits under step 4:
  \[(1 - p)^{n-m}\]
- Probability to improve exactly one of the wrong bits:
  \[mp(1 - p)^{m-1}\]
- This results in
  \[P\{\bar{a}_k' \text{ better than } \bar{a}_k\} \geq mp(1 - p)^{m-1}(1 - p)^{n-m} = mp(1 - p)^{n-1}\]
Analysis: Evolutionary Search

• Expected number of iterations until an improvement happens:

\[ E_{1\text{-bit-improv}} \leq \frac{1}{mp(1 - p)^{n-1}} \]

• Expected total number of improvements:

\[ E_{\text{iter}}(n) = \sum_{m=1}^{n} E_{1\text{-bit-improv}} \leq \frac{1}{p(1 - p)^{n-1}} \sum_{m=1}^{n} \frac{1}{m} \]

(Assuming we need \( n \) 1-bit improvements)

• Now we use:

\[ \lim_{n \to \infty} \left( \sum_{i=1}^{n} \frac{1}{i} - \ln n \right) = \gamma = 0.522... \]
Analysis: Evolutionary Search

- We finally get

\[ E_{iter}(n) \approx \frac{1}{p(1-p)^{n-1}} \ln(n) \]

- Assuming \( p = i/n \) with an integer \( i \):

\[ E_{iter}(n) \approx \frac{n}{i} e^i \ln n \]

- For \( p = 1/n \):

\[ E_{iter}(n) \approx ne \ln n \]
Concluding Remarks

• For this simple example:
  – Evolution-like algorithms are almost linear, not exponential, concerning their running time

• Analysis of algorithm 2 oversimplified:
  – Only one-bit mutations
  – Only improving mutations
  – Only an upper bound on $E(n)$
  – We can assume to start with $n/2$ correct bits

• Algorithm 2 is a so-called (1+1)-algorithm