Lecture 3

Evolutionary Algorithms

Invited Guest Professorship
Université Lois Pasteur, Strasbourg

Prof. Dr. Gisbert Schneider
Goethe-University, Frankfurt

25 November 2008, (c) G. Schneider
Optimization Tasks

Adaptive Walk in a “fitness landscape”
Types of fitness landscapes

The Dream

The Nightmare
Local Neighborhood Search

- fixed $N$
- variable ("adaptive") $N$
Local Neighborhood Strategies

Local (neighborhood) search

- Any-ascent/Stochastic hill-climbing
  - Simulated annealing
  - Threshold-accepting
- Steepest/First-ascent hill-climbing
- Population-based
  - Genetic algorithms
  - Evolution strategies
  - Evolutionary programming
- Tabu search
Elements of artificial adaptive systems (Koza)

- Structures that undergo adaptation
- Initial structures (starting solutions)
- Fitness measure that evaluates the structures
- Operations to modify the structures
- State (memory) of the system at each stage
- Method for designating a result
- Method for terminating the process
- Parameters that control the process
Learning from Nature ... 

- Genetic algorithms
- Evolution strategies
- Particle Swarm Optimization (PSO)
- Ant Colony Optimization (ACO)
Which search strategy should be applied?

Global & local search techniques guided by
- Random parameter variation
- Stepwise systematic parameter variation
- Gradients in the fitness landscape ...

There is no one best method.

Evolutionary algorithms are
- easily implemented
- robust
- able to cope with high-dimensional optimization tasks
Searching in real and artificial fitness landscapes

Quality (experimental)

"Fitness" (calculated)

Idealized path

Neural network model of peptide antibody recognition
a) Griewangk Funktion
\[ f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1 \]

b) Rastrigin Funktion
\[ f(x) = 10 \cdot n + \sum_{i=1}^{n} \left( x_i^2 - 10 \cdot \cos (2 \pi \cdot x_i) \right) \]

c) De Jong’s Sphere Funktion
\[ f(x) = \sum_{i=1}^{n} x_i^2 \]

d) Schaffer F6 Funktion
\[ f(x) = 0.5 + \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001 \cdot (x^2 + y^2))^2} \]

e) Rosenbrock Funktion
\[ f(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \]
Operators of Evolutionary Optimization

Variation

- Mutation
- Crossover
- Selection

Add new information to a population
Exploits information within a population
Genetic Algorithms: Chromosome Representation

<table>
<thead>
<tr>
<th>Descriptors</th>
<th>Real</th>
<th>Binary</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>clogP</td>
<td>5.19</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>cMW</td>
<td>404</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>O+N</td>
<td>5</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>H-donors</td>
<td>2</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>Surf\text{polar} [Å²]</td>
<td>59.9</td>
<td>010</td>
<td>011</td>
</tr>
</tbody>
</table>

1000111100001010
110010110001011

\textit{genes}\hspace{1cm}\textit{chromosome}
Principle of Evolutionary Searching

Step 1: Random search

Step 2 and following: “Local hill climbing”

Generate a **diverse** set of parameter values and hope for a hit

Generate **localized distributions** of parameter values and improve steadily

Adaptive neighborhood
The \((\mu, \lambda)\) Evolution Strategy

(Rechenberg, 1973)

1. **Start**
   - Create initial random population

2. **Evaluate fitness of each individual in population**

3. **Termination criterion satisfied?**
   - Yes: **End**
   - No: **Generate new population with \(\lambda\) children**

**Generation loop**
Algorithm of the (1,\(\lambda\)) Evolution Strategy

Initialize parent \((\xi^p, \sigma^p, Q^p)\);

For each generation:

- Generate \(\lambda\) variants \((\xi^v, \sigma^v, Q^v)\) around the parent:
  \[
  \sigma^v = \text{abs}(\sigma^p + G);
  \xi^v = \xi^p + \sigma^v \cdot G;
  \]
  calculate fitness \(Q^v\);
  Select best variant by \(Q^v\);
  Set \((\xi^p, \sigma^p, Q^p) \equiv (\xi^v, \sigma^v, Q^v)^\text{best}\);

Box-Muller: \[G(i, j) = \sqrt{-2\ln(i)} \cdot \sin(2\pi j)\] in \([-\infty, \infty]\)

\(i,j\): pseudo-random numbers in \([0,1]\)
Evolutionary Optimization: Protein backbone folding

Variables: Coordinates of “Beads on a string”
Fitness function: Empirical potential

„Folding“ of the ribonuclease A backbone

X-ray structure (1rtb)
Multiobjective Optimization (MO)

One possibility: weighted Fitness Function

\[ f(\text{property}) = w_1 \times \text{property}_1 + w_2 \times \text{property}_2 + \ldots \]

Disadvantages:

- The setting of the weights is **non-intuitive**
- Regions of the search space may be **excluded** due to the chosen weight setting
- Objectives may be **correlated**
- Only a **single solution** is found
MOGA – Multiobjective Genetic Algorithm

• Most optimization algorithms can only cope with a single objective
• Evolutionary Algorithms use a population of solutions

**MOGA** (Fonseca and Fleming 1993):

• Maps out the hypersurface in the search space where all solutions are tradeoffs between the different objectives
• Searches for a set of non-dominated solutions
• The ranking of solutions is based on dominance instead of a fitness function (**Pareto ranking**)
Multiobjective Optimization (MO)

- Many practical optimization applications are characterized by more than one objective.
- Optimal performance in one objective often correlates with low performance in another one.
- Strong need for a compromise.
- Search for solutions that offer acceptable performance in all objectives.
MOGA – Multiobjective Genetic Algorithm

Potential solutions in a two-objective ($f_1$ and $f_2$) minimization problem:

- Non-dominated solution (Pareto solution)
- Dominated solution
- Number of times the solution is dominated
Particle Swarm Optimization (PSO)

• Biological motivated optimization paradigm
• Individuals move through a fitness landscape
• Particles can change their movement direction and velocities
• “Social” interactions between individuals of a population
• Each particle knows about its own best position and all particles know about the overall best position
Particle Swarm Optimization (PSO) – cont.
PSO Algorithm

begin

initialize particle Positions and Memories
evaluate Fitness

while (stop criterion ≠ true)
    compute new Particle Velocities
    compute new Particle Positions
    evaluate Fitness
    update Memories
end

end
**PSO: Update Rule 1 (Standard Type)**

\[ v_i(t+1) = v_i(t) + 2 \cdot r_1 \cdot (p_i - x_i(t)) + 2 \cdot r_2 \cdot (p_b - x_i(t)) \]

- **v**: Velocity vector of a particle
- **x**: Position of a particle
- **i**: Dimension
- **t**: Epoch ("time")
- **r_1** and **r_2**: pseudo-random numbers between 0 and 1
- **p_i** and **p_b**: best Positions stored in the individual and social Memories
**PSO: Update Rule 2 (Inertia Weight Type)**

\[
v_{i}(t+1) = w \cdot v_{i}(t) + n_1 \cdot r_1 \cdot (p_i - x_i(t)) + n_2 \cdot r_2 \cdot (p_b - x_i(t))
\]

- **v**: Velocity vector of a particle
- **x**: Position of a particle
- **w**: Inertia weight
- **i**: Dimension
- **t**: Epoch ("time")
- **n_1** and **n_2**: Constants for the individual and social terms
- **r_1** and **r_2**: Pseudo-random numbers between 0 and 1
- **p_i** and **p_b**: Best Positions stored in the individual and social Memories

\[
w = w_{start} - \frac{w_{start} - w_{end}}{MaxEpochs} \cdot Epochs
\]
PSO: Individual and Social Memory

Convergence of PSO for the minimization of the Griewangk-Function after 100 epochs

Higher weight on the social memory

\[
\begin{align*}
n_1 &= 1 \\
n_2 &= 2 \\
\end{align*}
\]

\[
\begin{align*}
n_1 &= 2 \\
n_2 &= 1 \\
\end{align*}
\]
PSO: Update Rule 3 (Constriction Type)

\[ v_i(t+1) = K \cdot (v_i(t) + n_1 \cdot r_2 \cdot (p_i - x_i(t)) + n_2 \cdot r_2 \cdot (p_b - x_i(t))) \]

\( v \): Velocity vector of a particle
\( x \): Position of a particle
\( K \): Constriction constant
\( i \): Dimension
\( t \): Epoch (“time”)
\( n_1 \) and \( n_2 \): Constants for the individual and social terms
\( r_1 \) and \( r_2 \): Pseudo-random numbers between 0 and 1
\( p_i \) and \( p_b \): Best Positions stored in the individual and social memories

\[ K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \]

\( \varphi = n_1 + n_2, \varphi > 4 \)
The **Constriction constant** regulates swarm behavior in two ways (Kennedy & Eberhart, 2001):

- It leads to swarm convergence after a certain period of time.
- If the swarm has found a local optimum, the value of $K$ determines its ability to escape the local optimum.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{rosenbrock.png}
\caption{Rosenbrock function (after 100 epochs)}
\end{figure}

Constriction-Type \hspace{2cm} Standard-Type

\textit{(after 100 epochs)}
Particle Trajectories

- The particle visited several local optima before the global optimum was found
Exercise

1) Go to the modlab website: www.modlab.de software PSOViz
   • make yourself familiar with the PSOViz software options

2) Start PsoViz from your local computer (or directly from the website)

   Choose a fitness function and change the various PSO parameter settings.
   • Do you see an influence on the convergence behavior?
   • How do you explain the differences?
   • Do you observe „typical problems“ with PSO?