Global optimization and Evolutionary algorithms

Jost Neigenfind

Gute Ideen in der theoretischen Biologie/Systembiologie
July 3, 2007

Outline

- Motivation
- Differentiable target functions
- Non differentiable target functions
- Simulated annealing
- Evolutionary algorithms
- Gene expression programming

Motivation

Defining the target function

- target function: $f : \mathbb{R}^D \longrightarrow \mathbb{R}$
- $\text{argmin}_x f(x)$
- e.g. data fitting, maximizing likelihoods and other optimization problems

Some examples from real world

Non biological examples:

- chip design
- engeneering
- economics

Some examples from real world

In bioinformatics:

- Maximizing a likelihood function in ...
  - ... phylogeny
  - ... haplotyping
  - ... proteomics
- Minimizing the inner energy of molecules
- Linear and non linear regression (e.g. data fitting, micro array analysis)

Features of the target function

For each of the above mentioned problems target function can be defined with different features:

- target function
  - dimensionality
  - is differentiable or non differentiable
  - constraints exist
- function value
  - is continous or discrete
  - has more than one optimum

Optimization of differentiable target functions

How to find an optimum

- target function must be two times differentiable in every point
- 1. calculate Jacobian matrix (first partial differentiation)

Jacobian matrix

$$f'(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{bmatrix}$$

where x is a vector with D elements

How to find an optimum

- target function must be two times differentiable in every point
- 1. calculate Jacobian matrix (first partial differentiation)
- 2. calculate Hessian matrix (second partial differentiation)

Hessian matrix

$$f''(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial f}{\partial x_1 \partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_D \partial x_1} & \cdots & \partial x_D \partial x_D \end{bmatrix}$$

where x is a vector with D elements

How to find an optimum

- target function must be two times differentiable in every point
- 1. calculate Jacobian matrix (first partial differentiation)
- 2. calculate Hessian matrix (second partial differentiation)
- 3. if the target function is quadratic, the extrem values can be calculated directly

Since 3. is not alway true, one has to proceed interatively

- method of steepest descent is one of the simplest gradient based techniques
- f''^{-1} is replaced by the identity matrix
- the iteration looks as follows:
  $x_{n+1} = x_n - \gamma \cdot g(x_n)$ where γ defines a step size

Steepest descent

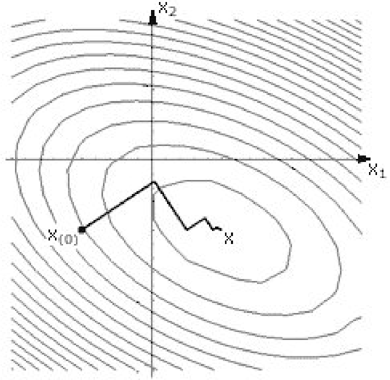


Figure: Path of the steepest descent method (http://www.basegroup.ru/images/neural/conjugate/pict2.gif)

Example

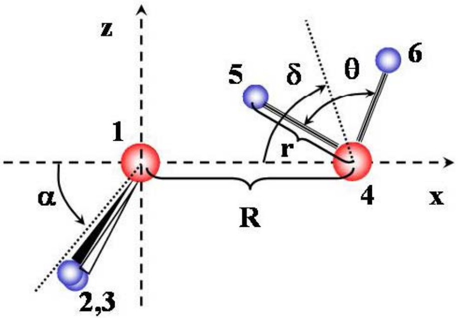


Figure: Minimizing the energy of two H_2O molecules (http://page.mi.fu-berlin.de/~burkhard/Lectures/Sim_Biomol_04/ue4.html)

There are more sophisticated methods of gradient based techniques

- Gauss-Newton
- Fletcher-Reeves
- Davidon-Fletcher-Powell
- Broyden-Fletcher-Goldfarb-Shanno
- Levenberg-Marquardt

Problem

- target function is not uni-modal
  $\Rightarrow$ likely that above mentioned methods will get caught in local optimum
- function value need not to be continously

Optimization of non differentiable target functions

Simpelx based methods

- Definition: an D dimensional simplex has D+1 affinely independent points in Euclidean space of dimension D or higher
- Example: Nelder-Mead method

Example of Nelder-Mead algorithm

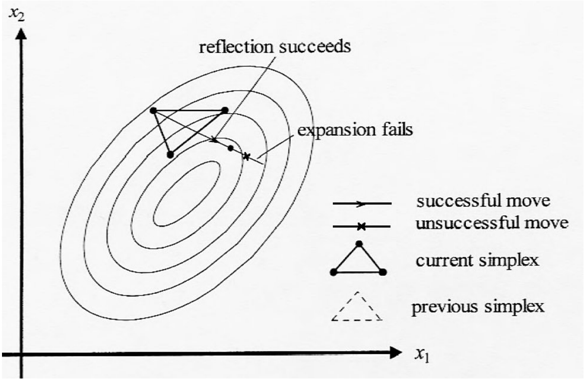


Figure: Evolution of the Nelder-Mead method (Differential Evolution, Storm Price)

Example of Nelder-Mead algorithm

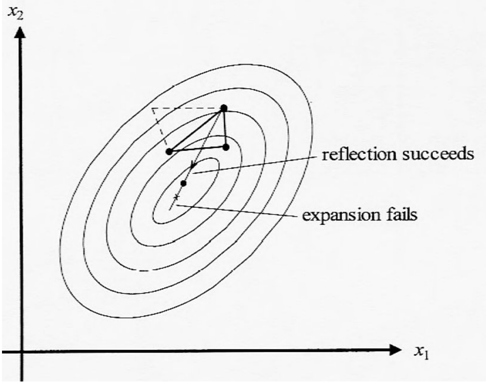


Figure: Evolution of the Nelder-Mead method (Differential Evolution, Storm Price)

Example of Nelder-Mead algorithm

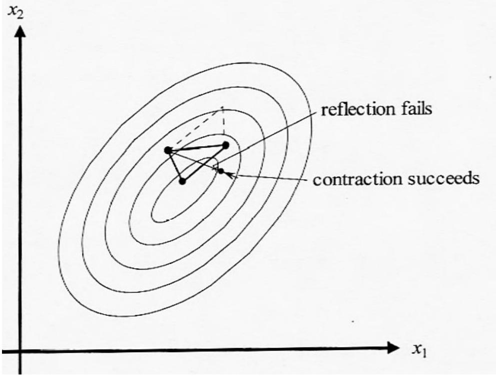


Figure: Evolution of the Nelder-Mead method (Differential Evolution, Storm Price)

Example of Nelder-Mead algorithm

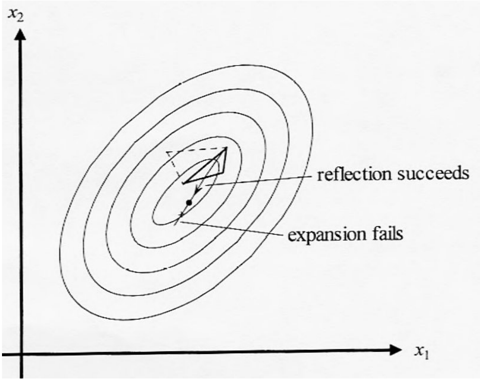


Figure: Evolution of the Nelder-Mead method (Differential Evolution, Storm Price)

Applet for Nelder-Mead method

http://de.wikipedia.org/wiki/Downhill-Simplex-Verfahren

Simulated annealing

The term "annealing" comes from metallurgy

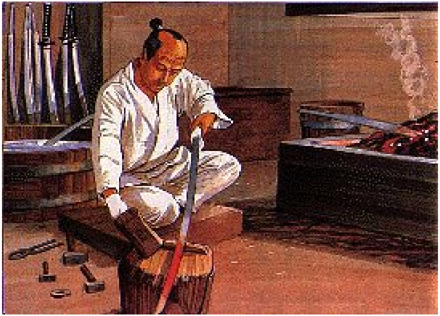


Figure: Japanese swordsmith

An only hardened sword is useless

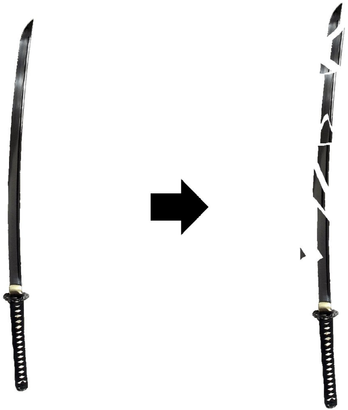


Figure: A sword will burst soon if it is only hardened

This burst is caused by the atomic structure of the sword

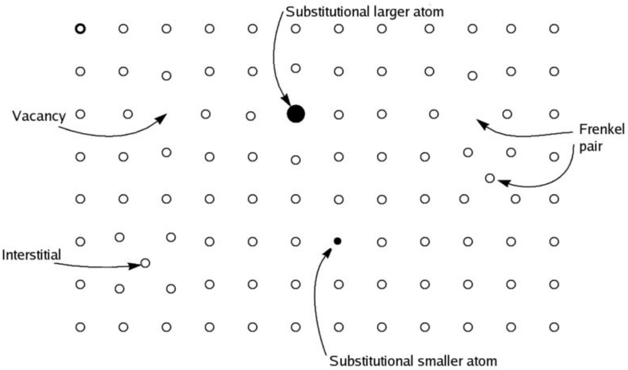


Figure: Defects of the atomic structure (http://en.wikipedia.org/wiki/Image:Defecttypes.png)

Annealing would result in:

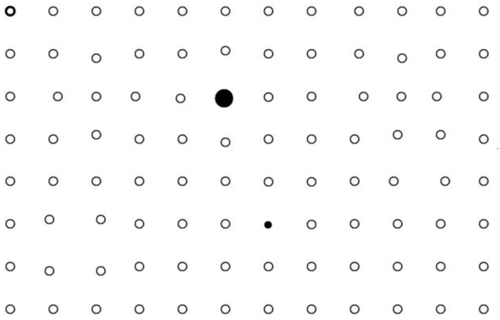


Figure: The atomic structure becomes more regular, crystal like

Annealing can be applied to optimization problems

- postions of atoms $\equiv$ parameters of a considered problem
- inner energy of atomic ensemble $\equiv$ target function
- temperature $\equiv$ parameter of probability function for accepting parameterizations resulting in worse function values

Formulation of simulated annealing

- for atoms: $P(\Delta E) \sim \exp\left(\frac{-\Delta E}{k \cdot T}\right)$
  where T represents the temperature, ΔE the energy difference of two states and k the Boltzman constant ($k \approx 1.381 \cdot 10^{23} \frac{J}{K}$)
- for problem function: $\Theta = \exp\left(-\frac{d}{\beta \cdot T}\right)$
  where T represents the "temperature", d the difference of two function values and β is a problem dependant control variable

Examples

- http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo41.php
- http://wwwai.wu-wien.ac.at/~hahsler/CPPAP/projekte/SS2001/Kammlander/

Application of simulated annealing to the TSP

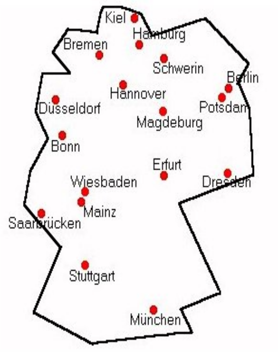


Figure: The Travelling Salesman Problem is NP hard (`http://www.f4.fhtw-berlin.de/~weberwu/diplom/tsp/HTMLS/SA_BSP.HTM`)

Application of simulated annealing to the TSP

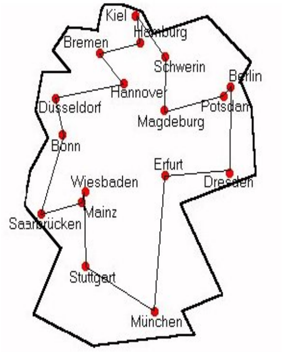


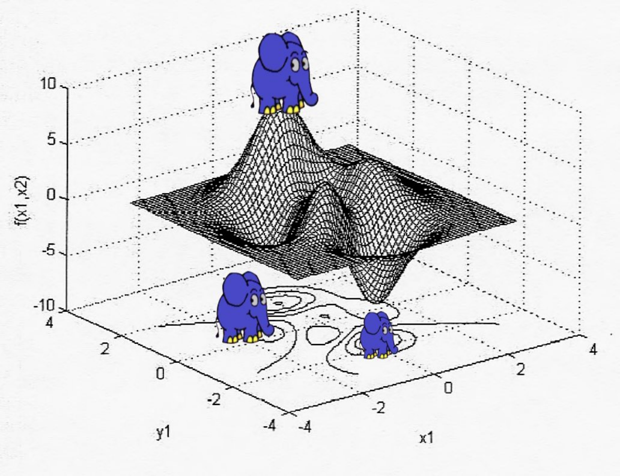
Figure: A solution of the given TSP (`http://www.f4.fhtw-berlin.de/~weberwu/diplom/tsp/HTMLS/SA_BSP.HTM`)

Example

http://www.heatonresearch.com/articles/64/page1.html

Evolutionary algorithms (EAs)

Inspiration



Evolutionary algorithms

- Evolution strategies (ESs)
- Genetic algorithms (GAs)

Evolution strategies and genetic algorithms

```
Initialization();       //choose starting population of μ members
while (not converged)   //decide the number of iterations
{
   for (i=0; i<λ; i++) //child vector generation: λ > μ
   {
     p1(i) = rand(μ);      //pick a random parent from μ parents
     p2(i) = rand(μ);      //pick another random parent  p2(i) != p1(i)
     c1(i) = recombine(p1(i),p2(i)); //recombine parents
     c1(i) = mutate(c1(i));          //mutate child
     save(c1(i));          //save child in an intermediate population
   }
   selection();            //μ new parents out of either λ, or λ+μ
}
```

Figure: Meta-algortihm for ESs and GAs (Differential Evolution, Storm Price)

Differences between ESs and GAs

	ES	GA
developed by	Rechenberg & Schwefel	Holland & Goldberg
kind of problem	continous target function	combinatorial problems
parameters	continous	discrete

Differential evolution

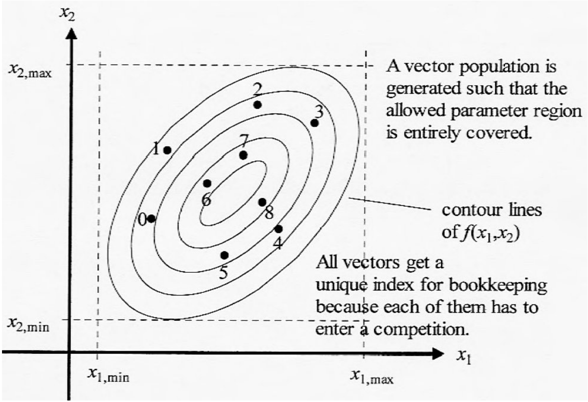


Figure: Evolution of the DE method (Differential Evolution, Storm Price)

Example for an ES: differential evolution

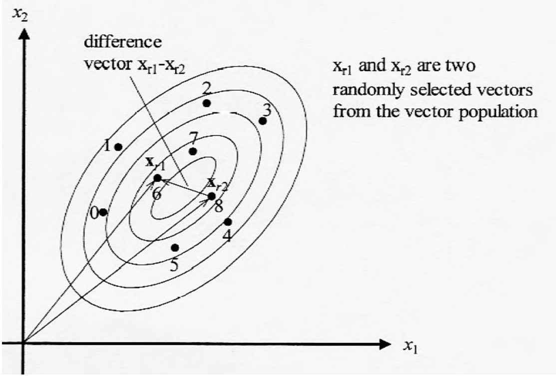


Figure: Evolution of the DE method (Differential Evolution, Storm Price)

Differential evolution

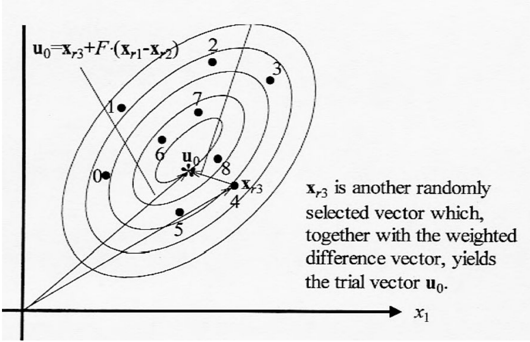


Figure: Evolution of the DE method (Differential Evolution, Storm Price)

Differential evolution

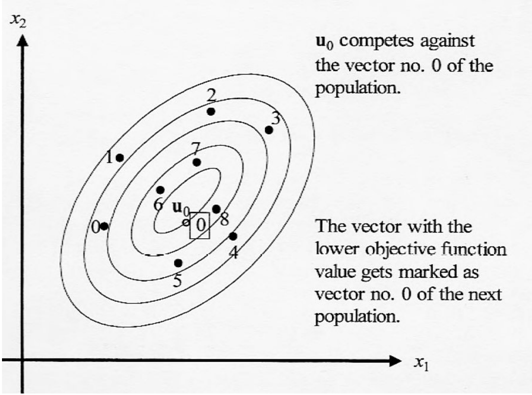


Figure: Evolution of the DE method (Differential Evolution, Storm Price)

Summary

	target f.		optimum			
method	continous	discret	local	global	# of f. values	t
steepest d.	yes	no	yes	no	single	fast
simplex	yes	yes	yes	no	single/multi	fast
simulated ann.	yes	yes	yes	yes*	single	slow
EAs	yes	yes	yes*	yes*	multi	slow

* but not guaranteed

Some tips for the optimization of a target function

- try as much algorithms as possible
- try different sets of parameters of the algorithm instead of default values
  - annealing schedule
  - population size ($\geq$ 10x number of parameters)
- start algorithm at different coordinates of solution space ($\geq$ 10x)

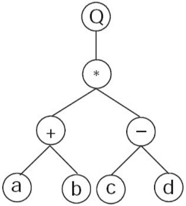
Gene expression programming

What is GEP

- fitting whole mathematical functions or programs to problem instances (similar to polynomial interpolation)
- functions are variated not parameters
- populations of functions are represented as kind of genes and chromosomes
- similar to the EAs, populations of functions compete against each other given a fitness function

Structure of a "gene"

The function $\sqrt{(a+b)\cdot(c-d)}$ can be represented by the following tree:



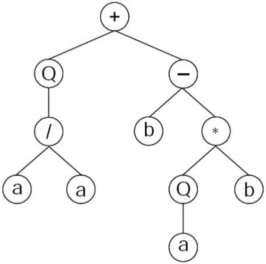
The sequence of the "gene" then is: $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ Q & \cdot & + & - & a & b & c & d \end{matrix}$

But ”gene”s are given a tail

The ”gene”:

```
012345678901234567890
+Q-/b*aaQbaabaabbaaab
```

defines the following tree and function respectively:

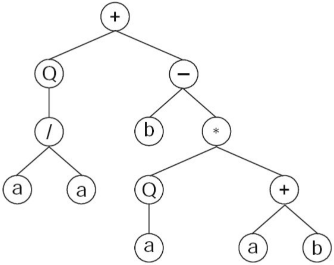


If now a mutation is introduced at position 9

The new "gene":

```
012345678901234567890
+Q-/b*aaQ+aabaabbaaab
```

defines the following new tree and function respectively:



⇒ tails are needed for mutations

C. Ferreira.
Gene expression programming: A new adaptive algorithm for solving problems.
Complex Systems, 13, 2:87–129, 2001.

k. V. Price, j. Lampinen, R. Storn.
Differential Evolution.
Springer.

Jr. M.P. Vecchi S. Kirkpatrick, C.D. Gelatt.
Optimization by simulated annealing.
Science, 220, 4598:671–680, 1983.

Thanks for your attention!

"Say...What's a mountain goat doing way up here in a cloud bank"

Figure: `http://www.geocities.com/francorbusetti/anneal.htm`