DM812
METAHEURISTICS

Lecture 13

Heuristic Methods for Continuous Optimization

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
<marco@imada.sdu.dk>

Outline

1. An Application Example in Econometrics

2. Heuristics for Continuos Optimization

Outline

1. An Application Example in Econometrics

2. Heuristics for Continuos Optimization

Capital Asset Pricing Model

Tool for pricing an individual asset i

Individual security's reward-to-risk ratio $= \beta_i \cdot$ Market's securities reward-to-risk ratio

$$\big(E(R_i) - R_f\big) = \beta_i \cdot \big(E(R_m) - R_f\big)$$

β_i sensitivity of the asset returns to market returns

Under normality assumption and least squares method:

$$\beta_i = \frac{\mathrm{Cov}(R_i, R_m)}{\mathrm{Var}(R_m)}$$

Alternatively:

$$R_{it} - R_{ft} = \beta_0 + \beta_1 \cdot (R_{mt} - R_{ft})$$

Use more robust techniques than least squares to determine β_0 and β_1
[Winker, Lyra, Sharpe, 2008]

Least Median of Squares

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t$$

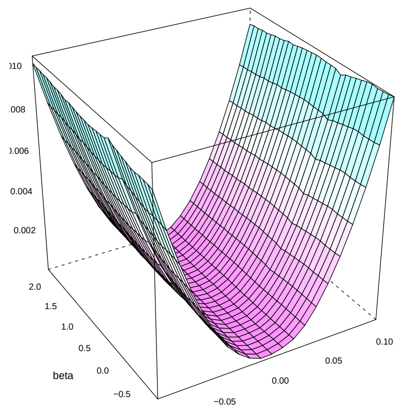
$$\epsilon_t^2 = \left(Y_t - \beta_0 - \beta_1 X_t\right)^2$$

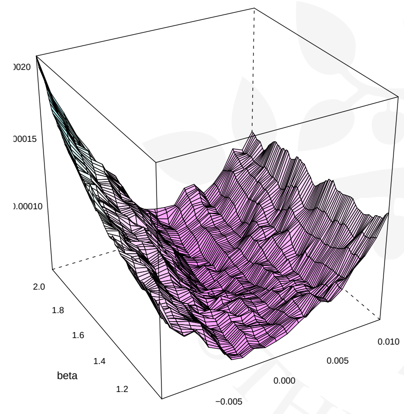
least squares method:

$$\min \sum_{t=1}^{n} \epsilon_t^2$$

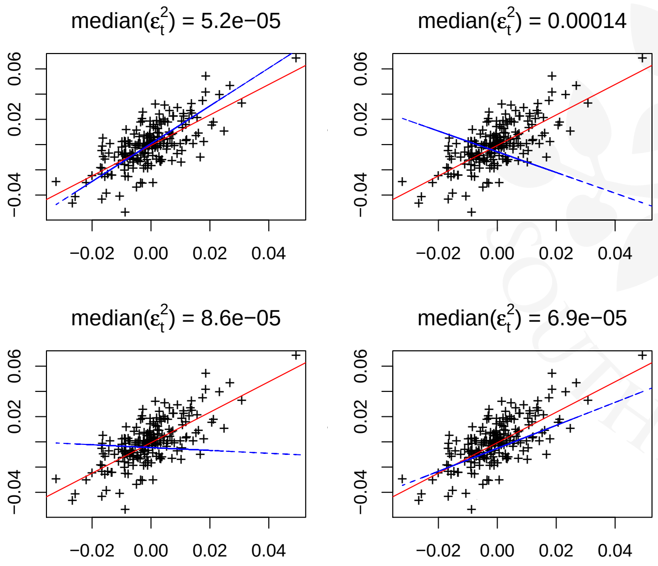
least median of squares method:

$$\min \left\{ \text{median} \left[\epsilon_t^2\right] \right\}$$





Four solutions corresponding to four different local optima
(red line: least squares; blue line: least median of squares)



Outline

1. An Application Example in Econometrics
2. Heuristics for Continuos Optimization

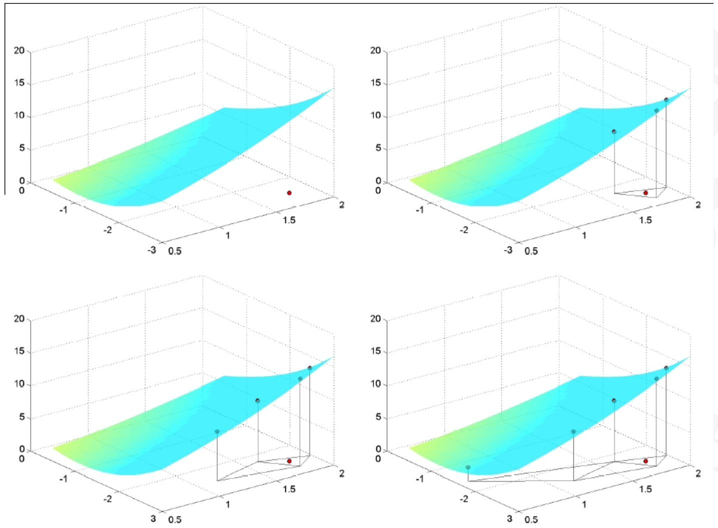
Optimization Heuristics

- Nelder-Mead
- Simulated Annealing
- Differential Evolution
- Particle Swarm Optimization
- Genetic Algorithm
- Ant Colony Optimization

Nelder-Mead

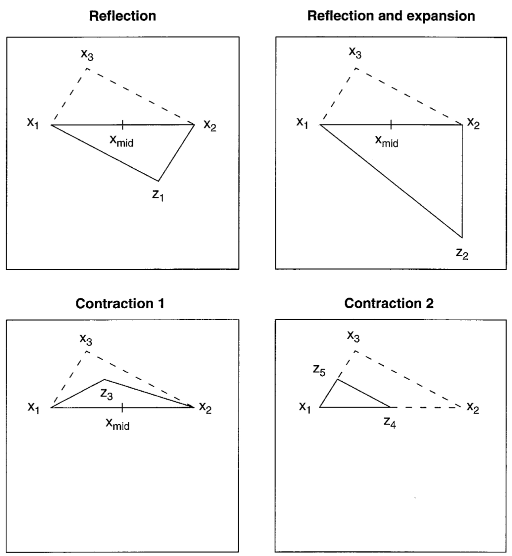
An Example
Heuristics for Continuos Optimization

Simplex based method [Spendley et al. (1962)]



Nelder-Mead (cont.)

An Example
Heuristics for Continuos Optimization



Nelder-Mead (cont.)

An Example
Heuristics for Continuos Optimization

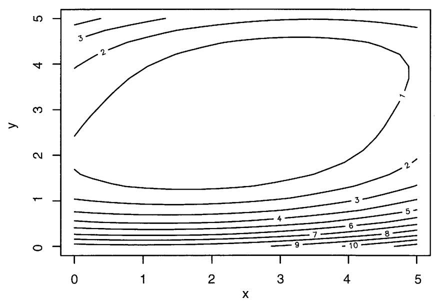
Nelder-Mead simplex method [Nelder and Mead, 1965]:

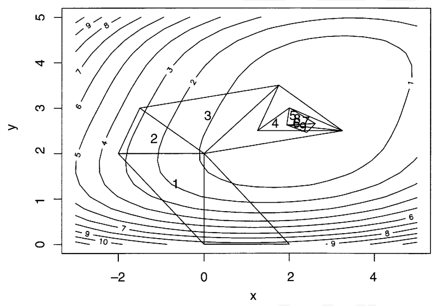
Algorithm 5 Simplex search.

```
Construct vertices x^(1),...,x^(n+1) of starting simplex
repeat
    Rename vertices such that f(x^(1)) ≤ ··· ≤ f(x^(n+1))
    if f(x^R) < f(x^(1)) then
        if f(x^E) < f(x^R) then x* = x^E else x* = x^R
    else
        if f(x^R) < f(x^(n)) then
            x* = x^R
        else
            if f(x^R) < f(x^(n+1)) then
                if f(x^O) < f(x^(n+1)) then x* = x^O else shrink
            else
                if f(x^I) < f(x^(n+1)) then x* = x^I else shrink
            end if
        end if
    end if
    if not shrink then x^(n+1) = x* (Replace worst vertex by x*)
until stopping criteria verified
```

Nelder-Mead (cont.)

Example:





Simulated Annealing

Simulated Annealing (SA):
determine initial candidate solution s
set initial temperature $T = T_0$
while termination condition is not satisfied **do**
  while keep T constant, that is, T_{max} iterations not elapsed **do**
    probabilistically choose a neighbor s' of s
      using proposal mechanism
    accept s' as new search position with probability:

$$p(T, s, s') := \begin{cases} 1 & \text{if } f(s') \leq f(s) \\ \exp \frac{f(s)-f(s')}{T} & \text{otherwise} \end{cases}$$

    update T according to annealing schedule

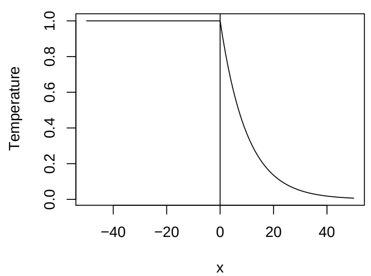
Proposal mechanism
The next candidate point is generated from a Gaussian Markov kernel with scale proportional to the actual temperature.

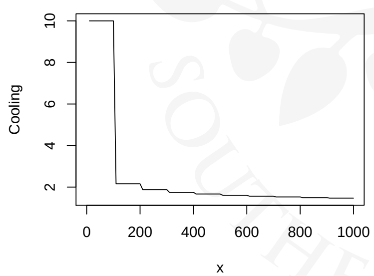
Simulated Annealing

Annealing schedule

- logarithmic cooling schedule [Belisle (1992)]

$$T = \frac{T_0}{\ln(\lfloor \frac{i-1}{I_{max}} \rfloor I_{max} + e)}$$



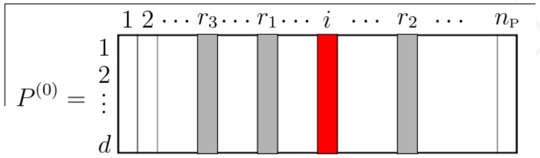


- threshold accepting [Dueck and Scheuer (1990)]
  accept if $\Delta < \tau$

Differential Evolution

Differential Evolution (DE)
determine initial population P
while termination criterion is not satisfied **do**
  for each solution x of P **do**
    generate solution u from three solutions of P by mutation
    generate solution v from u by recombination with solution x
    select between x and v solutions

Differential Evolution (cont.)



- Solution representation: $x = (x_1, x_2, \ldots, x_p)$
- Mutation:

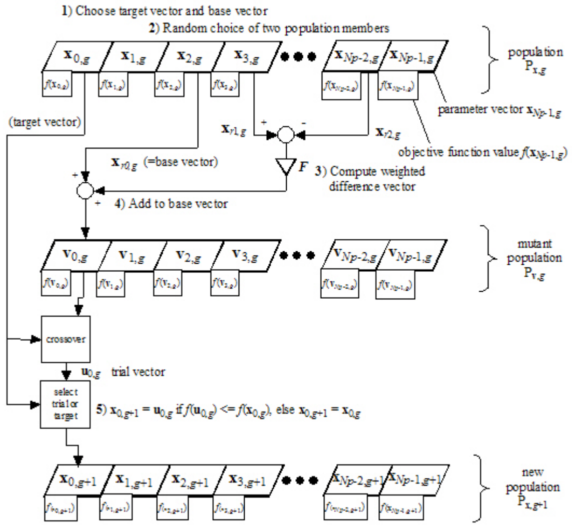
$$\mathbf{u} = \mathbf{r}_1 + F \cdot (\mathbf{r}_2 - \mathbf{r}_3) \qquad F \in [0, 2] \text{ and } (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \in P$$

- Recombination:

$$v_j = \begin{cases} u_j & \text{if } p < CR \text{ or } j = r \\ x_j & \text{otherwise} \end{cases} \qquad j = 1, 2, \ldots, p$$

- Selection: replace x with v if $f(v)$ is better

Differential Evolution (cont.)



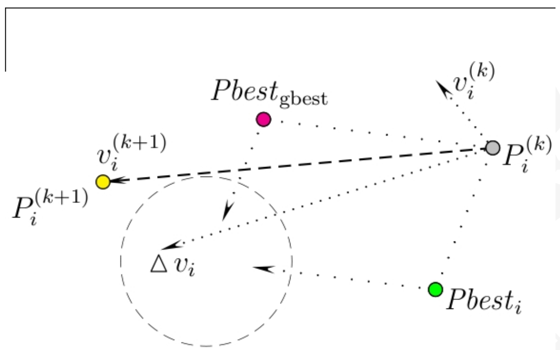
[http://www.icsi.berkeley.edu/~storn/code.html
K. Price and R. Storn, 1995]

Particle Swarm Optimization

Particle Swarm Optimization

Initialize parameters n_P, n_G and c
Initialize particles $P_i^{(0)}$ and velocity $v_i^{(0)}$, $i = 1, \ldots, n_P$
Evaluate objective function $F_i = f(P_i^{(0)})$, $i = 1, \ldots, n_P$
$Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \mathrm{argmin}_i(F_i)$
for $k = 1$ to n_G **do**
  for $i = 1$ to n_P **do**
    $\triangle v_i = c\,u\,(Pbest_i - P_i^{(k-1)}) + c\,u\,(Pbest_{\mathrm{gbest}} - P_i^{(k-1)})$
    $v_i^{(k)} = v^{(k-1)} + \triangle v_i$
    $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$
  end for
  Evaluate objective function $F_i = f(P_i^{(k)})$, $i = 1, \ldots, n_P$
  for $i = 1$ to n_P **do**
    if $F_i < Fbest_i$ **then** $Pbest_i = P_i^{(k)}$ and $Fbest_i = F_i$
    if $F_i < Gbest$ **then** $Gbest = F_i$ and $gbest = i$
  end for
end for

Particle Swarm Optimization

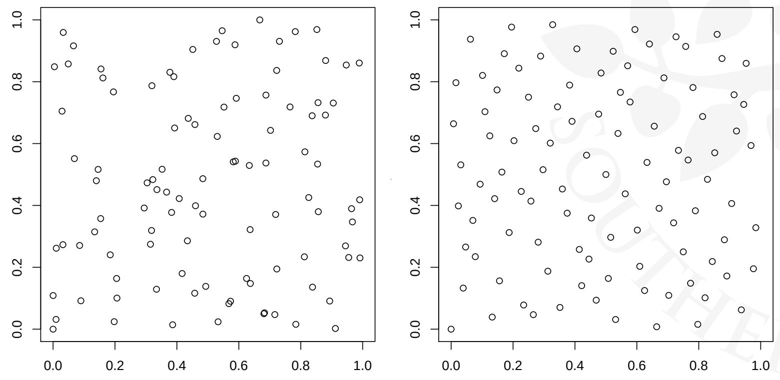


Generation of Initial Solutions

Point generators:

Left: Uniform random distribution (pseudo random number generator)

Right: Quasi-Monte Carlo method: low discrepancy sequence generator
[Bratley, Fox and Niederreiter, 1994]



(for other methods see *spatial point process from spatial statistics*)