Short Reminder of Nonlinear Programming

Kaisa Miettinen

Dept. of Math. Inf. Tech.

Email: kaisa.miettinen@jyu.fi

Homepage:
http://www.mit.jyu.fi/miettine

Contents

- Background
- General overview
  - briefly theory
  - some methods (no details)
  - different methods for different problems: onion-like structure
  - further references
- Then: multiobjective optimization
  - General overview
  - Concepts and some theory
  - Classification of methods
  - Applications
  - Further references

Some History

- Optimization natural – best possible
- Queen Dido – skin of ox – half circle and sea
- Long history in systems of equations, development of algebra, number theory,numerical mathematics
- Little attention to systems of inequalities – few exceptions
- Optimization flourishes when one has to do better than the others: war and economic competition
- 1940’s, war and industrialization
- Danzig/Pentagon, Kantorovich/Soviet Union
- Old mathematical results were rediscovered
- Simplex 1947, computers
- Background in applications, not theory

More History

- Optimization is an essential part in mathematics, engineering and business management
- Simplex made possible to solve large problems (transportation, scheduling, resource allocation etc.)
- Computers essential
- Nobel 1975 in economics (Kantorovich, Koopmans)
- Danzig included a well defined objective function in the model
- Nonlinear programming (numerical mathematics)
- Combinatorics, integer (discrete mathematics)

More History II

- Simplex ⇒ linear programming in 1947
- Karush-Kuhn-Tucker optimality conditions ⇒ nonlinear programming 1951
- Commercial applications (petrochemistry 1952)
- First commercial LP solver 1954
- Large-scale problems 1955
- Stochastic programming 1955
- Integer programming 1958
- Program = planning, ordering in the military (computer program was called code)

History of Mathematical Programming ed. by Lenstra et al., 1991, North-Holland

Optimization

- Optimal = best possible.
- What is best?
  - it depends…
- Applications everywhere.
- Function(s) to be minimized/ maximized.
- Varying variable values.
- Subject to constraints.

- Designing bridges, wings of aeroplanes
- Transporting timber
- Blending sausages
- Planning production systems
- Locating dumping places
- Scheduling projects
- Buying a car/ a house
- Finding shortest routes
- Etc.

Different Optimization Problems

- Linear
- Integer
- Mixed integer
- Combinatorial
- Quadratic
- Geometric
- Nonlinear

- Nondifferentiable
- Global
- Dynamic
- Stochastic
- Fuzzy
- Multiobjective

Background

- Trial and error not enough
- Mathematical model of the phenomenon
- Simulation possible – not enough
- Optimization needed
- **But: what is optimal – best possible?**
- ***Material e.g.: Bazaraa, Sherali, Shetty: Nonlinear Programming: Theory and Algorithms, John Wiley & Sons, 2nd or 3rd ed., Juha Haataja: Optimointitehtävien ratkaiseminen, CSC, 2004***

Nonlinear Programming

- Problem formulation $\quad \begin{array}{l} \text{minimize } f(x) \\ \text{subject to } x \in S, \end{array} \quad (1)$

  where

  f: $R^n \rightarrow R$ is an objective function,

  $x \in R^n$ is a decision variable vector and

  $S \subset R^n$ is a feasible region defined by constraint functions $S = \{x \in R^n \mid g_i(x) \leq 0, i=1,\dots,m, h_j(x) = 0, j=1,\dots,l \text{ and } x^l \leq x \leq x^u\}$
- Deterministic – stochastic
- Continuous – discrete
- Nonlinear – linear
- Convex - nonconvex

→ Continuous nonlinear programming

Optimality

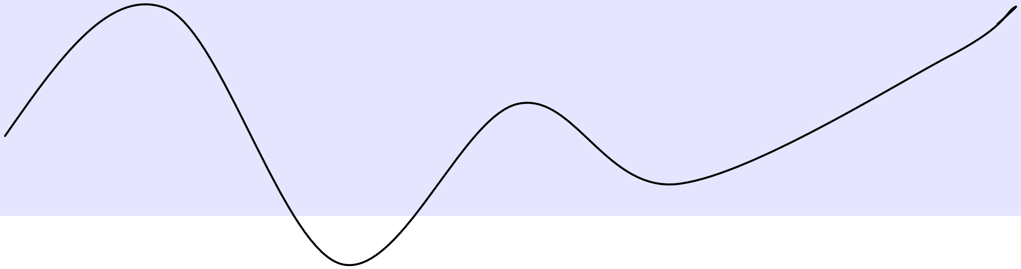
- **Definition: A point x* is the globally optimal solution of problem (1) if**

  $f(x^*) \le f(x)$ for all $x \in S$.

- **Definition: A point x* is the locally optimal solution of problem (1) if there exists $\delta > 0$ such that**

  $f(x^*) \le f(x)$ for all $x \in S$ such that $\|x-x^*\| \le \delta$.

- **Then, f(x*) is the optimum of (1).**



Kaisa Miettinen

Optimality, cont.

- Weierstrass theorem: Let S be a nonempty and compact (i.e. closed and bounded) set and let $f: S \rightarrow R^n$ be continuous in S. Then, there exists an optimal solution to (1) in S.

Concepts

- Let all the functions be continuous.
- If f is differentiable at x, we have gradient of f at x: $\nabla f(x) \in R^n$. Partial derivatives $\nabla f(x) = (\partial f(x)/\partial x_1, \ldots , \partial f(x)/\partial x_\nu)^T$.
- If f is twice differentiable, we have Hessian H(x).
- Set S is convex if $\forall$ x,y $\in$ S and $\lambda \in [0, 1]$ is valid $\lambda x+(1-\lambda)y \in S$. Line connecting any two points in S belongs to S. Intersection of convex sets is convex.
- Vectors $x^1, \ldots , x^p \neq 0$ are linearly independent if $\sum_{i=1}^p \lambda_i x^i = 0$ only for $\lambda_1 = \ldots = \lambda_p = 0$.
- Let $S \subset R^n$ be convex. $f: S \rightarrow R$ is convex if $\forall$ x,y $\in$ S and $\forall$ $\lambda \in [0, 1]$ is valid $f(\lambda x+(1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$.
  - $f: [a, b] \rightarrow R$ is convex if f' is increasing/ iff f'' is nonnegative.
  - sum of convex functions is convex. Convex function is continuous but not necessarily differentiable.

Concepts, cont.

- Let $\emptyset \neq S \subset R^n$ be a closed, convex set. Vector $0 \neq d \in R^n$ is a direction of S if $\forall\ x \in S$ is valid $x + \lambda d \in S\ \forall\ \lambda \geq 0$. Directions $d^1, d^2 \in S$ are separate if $d^1 \neq \alpha d^2$ for any $\alpha > 0$.
- $d \in R^n$ is a feasible direction at point $x \in S$, if $\exists\ \alpha^* > 0$ such that $x + \alpha d \in S\ \forall\ \alpha \in [0, \alpha^*]$.
- Let A be a symmetric n × n matrix. Directions $d^1, \ldots, d^p$ are conjugate if they are linearly independent and if $(d^i)^T A d^j = 0\ \forall\ i,j = 1, \ldots, p, i \neq j$.

Solving Optimization Problems

- Solution methods are usually iterative processes where a **series of solutions** is generated based on pre-determined instructions and guidelines.
- It is important when to stop the solution process.
- Algorithms for different problem types under different assumptions (differentiability, convexity)
- Sometimes it is important to scale the constraints or the variables so that the ranges of the values of the variables are comparable. This makes numerical calculation easier and improves the accuracy of the solution.

Problem types

- Single variable
- Unconstrained problems with several variables
  - gradient-free methods
  - gradient-based methods
- Constrained problems
  - penalty functions
  - direct methods
- Nondifferentiable problems
- Multiobjective optimization problems
- Global optimization
  - deterministic methods
  - stochastic methods
    - metaheuristics

Single Variable

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in [a,b] \end{array}$$

- Minimum: interior point with zero-gradient, point with no gradient or end point of interval
- Theorem: Let f: R → R be continuously differentiable. If x* is the minimal solution, then f'(x*)=0 and f''(x*) > 0.
- Methods based on the assumption: f is unimodal.
- f is unimodal on [a,b] if for some point x* ∈ (a,b) is valid: f is strictly decreasing on [a,x*) and strictly increasing on (x*,b].

Single variable cont.

- **f unimodal, shorten the interval: Select points x^1 and x^2 such that $a \leq x^1 \leq x^2 \leq b$.**
  - **if $f(x^1) < f(x^2)$, study $[a,x^2)$.**
  - **if $f(x^1) > f(x^2)$, study $(x^1, b]$.**
  - **if $f(x^1) = f(x^2)$, study (x^1,x^2).**
- **Methods**
  - **elimination methods**
    - **e.g. golden section**
  - **interpolation methods**
    - **use polynomials and derivatives**

Unconstrained, several variables

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbf{R}^n \end{array}$$

- Optimality conditions: Let x ∈ R^n be given. Is it is locally or globally minimal?
- Conditions satisfied when it is optimal are **necessary** optimality conditions. They often necessitate differentiability.
- Conditions that guarantee optimality are **sufficient** optimality conditions.

Necessary Conditions

- Definition: Let f: $R^n \rightarrow R$. $d \in R^n$ is its descent direction at x^*, if $\exists\ \delta > 0$ such that $f(x^* + \lambda d) < f(x^*)\ \forall\ \lambda \in (0, \delta]$.
- Theorem: Let f be differentiable at x^*. If $\exists$ d such that $\nabla f(x^*)^T d < 0$, then d is a descent direction of f at x^*.
- Theorem: (1st order necessary opt. cond.) Let f be differentiable at x^*. If x^* is locally minimal, then $\nabla f(x^*) = 0$, i.e., x^* is a critical point.
- Theorem: (2nd order necessary opt. cond.) Let f be twice differentiable at x^*. If x^* is locally minimal, then $\nabla f(x^*) = 0$ and Hessian $H(x^*)$ is positive semidefinite.
- Necessary condition does not imply optimality (saddle point)

Sufficient conditions etc.

- Let f be twice differentiable at x^*. If $\nabla f(x^*)=0$ ja Hessian $H(x^*)$ is positive definite, then x^* is strictly locally minimal.
- Let f be convex and differentiable. Then x^* is globally minimal iff $\nabla f(x^*) = 0$.
- Let f be convex. Then any locally minimal solution is globally minimal. (No differentiability assumed!)

Basic Ideas in Methods

- **Direct search methods**
- **Methods using differentiability**
- **Iterative basic algorithm for $x^h \rightarrow x^*$ as $h \rightarrow 1$, i.e., $\lim||x^{h-}x^*|| = 0$ such that $f(x^{h+1}) < f(x^h)$.**
  1) **Select starting point x^0. Set $h=0$.**
  2) **Generate a (descent) search direction d^h.**
  3) **Calculate step length λ_h.**
  4) **Set $x^{h+1} = x^h+\lambda_h d^h$.**
  5) **Stop if stopping criterion ok. Otherwise, set $h=h+1$ and go to 2).**
- **Crucial steps 2) and 3)**
- **Different rates of convergence**
  - **linear, superlinear, quadratic**
  - **local and global**

Direct Search Methods

- Arbitrary descent directions
- Coordinate-wise, univariate search
  - easy but slow, may fail
- Hooke and Jeeves
  - univariate search + pattern search (fixed or not)
- Powell
  - most developed pattern search method
  - can compete with methods using derivatives but no linearly dependent search directions allowed
  - one coordinate directions is replaced by pattern search direction
  - quadratic f, n conjugate search directions, minimum in n iterations (near minimum, 2nd order polynomial approximates well twice differentiable function)
- Simplex or Nelder Mead method

Gradient based methods

- Use objective function values + derivatives
  - possible difference approximations
- Descent methods: $x^{h+1}=x^h+\lambda_h d^h$, $f(x^{h+1})<f(x^h)$, $h=1,2,\ldots$
- Steepest descent method
  - $d^h = -\nabla f(x^h)$
  - sic-sac phenomenon (perpendicular directions)
  - stopping criteria $\max_{i=1, \ldots n} |\partial f/\partial x_i|<\varepsilon$, $\|\nabla f(x^{h+1})\|^2<\varepsilon$, $\|f(x^{h+1})-f(x^h)\|<\varepsilon$, $\|x^{h+1}-x^h\|<\varepsilon$.
  - global but linear convergence
  - sensitive to scaling of variables

Newton’s method

- $d^h = -H(x^h)^{-1}\nabla f(x^h)$, fixed step or not
- inverse matrix – solve system of equations
- second order information improves convergence
- stopping criterion: $\|d^h\|$ small enough, $\|\nabla f(x^{h+1})\|^2 < \varepsilon$,
- quadratic but local convergence
- d^h not necessarily descent if Hessian not pos. def.
- diff. approx. ok
- approximate Hessian – quasi-Newton methods

Property	**Steep.**	**Newton**
descent d	yes	H pos.def.
global conv	yes	no
local conv.	sic-sac	yes
bottleneck	λ_h	$H(x^h)^{-1}$
quadr. prob	h ?	1 iter.

Quasi-Newton method

- Variable metric method
- Inverse of Hessian approximated by a symmetric pos. def. matrix
- $d^h = -D^h \nabla f(x^h)$ - d^h is descent when $\nabla f(x^h) \neq 0$
- Davidon-Fletcher-Powell
- Broyden-Fletcher-Goldfarb-Shanno
- usually local and superlinear convergence
- When n increases, hard to handle matrices
- Difference approximations may disturb

Conjugate Gradient Method

- Originally, for solving linear systems of equations
- Less efficient than quasi-Newton
- Less requirements for memory – for large problems
- Improve gradient direction – add positive multiple of earlier search directions
- E.g. Fletcher and Reeves

Summary: Unconstrained Methods

- Efficiency
  - Newton
  - quasi-Newton
  - conjugate gradient
  - direct search
- Direct search simple but slow
- Derivative-based method usually converge to the closest minimum but direct search methods may (with good luck) find a global minimum in case of a non-unimodal function

Gradients

- Analytic formulas
- Symbolic differentiation
- Automatic differentiation
- Difference approximations
  - Forward
  - Backward
  - Central

Constrained optimization

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m, \\ & h_i(\mathbf{x}) = \mathbf{0}, \quad \mathbf{i} = \mathbf{1}, ..., \mathbf{l}. \end{array}$$

- Let S be the feasible region, D a cone of feasible directions and F a cone of decreasing directions
- If x^* is locally minimal and if $\nabla f(x^*)^T d < 0$, then d is not in D. Thus, we have a necessary condition for local optimality: none if the decreasing directions can be feasible.

Karush-Kuhn-Tucker

- Assume regularity of constraint, i.e., constraint qualification (many types), like gradients of active inequality constraints and equation constraints are linearly independent
- Necessary condition: Let f, g_i and h_i be continuously differentiable at a regular point $x^* \in S$. If x^* is locally minimal, $\exists\ \mu_i \geq 0$ (i=1, ..., m) and ν_i (i=1, ... , l) such that

  (1) $\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{i=1}^{l} \nu_i \nabla h_i(x^*) = 0$

  (2) $\mu_i g_i(x^*) = 0\ \forall$ i=1,... , m.
- Sufficient condition: Let f, g_i and h_i be cont. diff. & convex. We study $x^* \in S$. If $\exists\ \mu_i \geq 0$ (i=1,..., m) and ν_i (i=1,..., l) s.t.

  (1) $\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{i=1}^{l} \nu_i \nabla h_i(x^*) = 0$

  (2)$\mu_i g_i(x^*) = 0\ \forall$ i=1,..., m, then x^* is globally minimal
- Hard to use optimality conditions directly -> methods

Methods for Constrained Problems

- Special structure: constraints linear equations, linear inequalities or nonlinear equations, objective function quadratic, etc.
- Indirect methods
  - convert the problem into a sequence of unconstrained problems by shifting the constraints into objective function or variables
  - penalty and barrier methods
  - methods utilizing Lagrangian
- Direct methods
  - take constraints explicitly into account
  - stay in the feasible region
  - projected gradient
  - active set methods

Indirect Methods

- Shift the variables so that constraints are automatically valid
- Penalty and barrier functions
  - series of unconstrained problems
  - punish infeasible solutions - add a penalty term to the obj.function
  - penalty: generate a series of infeasible solutions converging to optimum
  - $f(x) + r^h(\sum_{i=1}^{m} (\max[0, g_i(x)])^p + \sum_{i=1}^{l} |h_i(x)|^p)$, with p¸ 2.
  - increase r^h (approaches infinity)
  - barrier: add a term to obj.function that hinders points from leaving S. Generates a sequence of feasible points converging to optimum (only inequality constraints)
  - $f(x) + r^h(\sum_{i=1}^{m} (-1/g_i(x)))$
  - decrease r^h (approaches zero)
  - if r^h changes too slowly, many problems, if fast, hard minimizations
  - exact: nondifferentiable, finite r^h

Indirect Methods cont.

- Augmented Lagrangian
  - equation-type constraints
  - combination of Lagrangian and penalty term
  - $f(x) + \sum_{i=1}^{l} \nu_i h_i(x) + 1/2\rho\sum_{i=1}^{l} (h_i(x))^2, \rho > 0$
  - ν should approach Lagrangian multipliers
  - ρ increases
  - problem is differentiable
  - minimum can be found with finite values of ρ

Direct Methods

- Methods approximating constraints
  - linearize constraints near some point and use LP (e.g. cutting plane method)
- Methods of feasible directions
  - search direction improving and feasible
  - method of projected gradient
    - project –rf(x) to be feasible (projection matrix)
  - active set methods
  - generalized reduced gradient
    - decrease number of variables
  - SQP
    - efficient, series of quadratic approximations

Local/Global Optimality

- Now we can find local minimum and maximum points of a differentiable function defined in an open set. How about global optimality?
- If function is concave, maximum is global and if function is convex, minimum is global
- Let f be a C^2-function defined in a convex set U
  - If f is concave and x* is a critical point, then x* is a global maximum point
  - If f convex and x* is a critical point, then x* is a global minimum point
- Checking convexity/concavity can be done with the Hessian matrix (semidefiniteness in U) (negative definite -> max, positive definite -> min)

Global Optimization

- If problem is nonconvex, above-mentioned methods find one of the local minima (closest to starting point)
- Guaranteeing global optimality – hard
- Both reliability and efficiency important; conflicting
- Find solution close to x* in sets

$$\{\mathbf{x} \in S \mid \|\mathbf{x} - \mathbf{x}^*\| \leq \varepsilon_{\mathbf{x}}\}$$

$$\{\mathbf{x} \in S \mid f(\mathbf{x}) - f(\mathbf{x}^*) \leq \varepsilon_f\}.$$

Global Optimization, cont.

- Deterministic methods
  - branch and bound
  - assume special properties (Lipschitz continuity)
  - form new problems that are even more difficult
- Stochastic methods
  - use random points
  - multistart (random points & local optimization)
  - clustering (few iterations of local search)
  - metaheuristics
    - simulated annealing
    - genetic algorithms

Simulated Annealing

- Accept as search directions
  - decreasing ones
  - also directions of deterioration with some probability that decreases (Metropolis criterion)
  - physical analogy (anneal material and let it cool down slowly to reach its energy minimum)
  - different implementations
- Widely used in combinatorial problems

One Algorithm for SA

- Current iteration point $x \in R^n$. Next candidate solution y is generated by varying one component i of x at a time, i.e., $y_i = x_i + q\, d_i$,

  where q is a uniformly distributed random number from [-1,1] and $d \in R^n$ is a search direction.
- y is accepted if $f(y) < f(x)$ or $e^{(f(x)-f(y))/t} > p$, where p is a uniformly distributed random number from [0,1].
- Here, $t>0$ is a temperature and it is decreased during the algorithm.

Genetic/Evolutionary Algorithms

- Simulate evolution process
- Population-based
- Coding (binary, gray, real)
- Many different variants/implementations (differetial evolution, island model, …)
- Does not require continuity (or differentiability)
- Can handle multimodality/nonconvexity
- Often time-consuming
- May work when other methods do not

Genetic/Evol. Algorithms cont.

- Selection
  - tournament
  - roulette-wheel
- Crossover
  - single-point
  - uniform
  - arithmetic
  - heuristic $y = r\ (x^2 - x^1) + x^{2)}$
- Mutation
  - normally distributed
  - random
  - non-uniform
  - polynomial

Genetic/Evol. Algorithms, cont.

1. Set population size, tournament size, crossover rate, mutation rate and elitism size. Set the parameters of the stopping criterion.
2. Initialize the population with random numbers.
3. Compute the fitness function values. Perform selection, crossover, mutation and elitism in order to create a new population.
4. If the stopping criterion is not satisfied, return to step 3. Otherwise, choose the best individual found as the final solution.

- Stop: max number of generations, difference between best results of fixed number of generations.

GA + Constraint Handling

- As such, genetic algorithms are not good at handling constraints efficiently.
- Ideas based on classical penalty functions and modifications in population-based environments.
- New modifications and comparisons.
- Some emphasis on stopping criteria.
- Feasibility may be taken into account in selection

Method of Superiority of Feasible Points

- New fitness function
- Feasible individuals have better fitness function values than the ones outside the feasible region

$$f^{\circ}(x) = f(x) + r\left(\sum_{j=1}^{m} \mathrm{max}\left[0, g_j(x)\right]\right) + \theta_i(x)$$

$$\theta_i(x) = \begin{cases} 0 & \text{if } X^i \cap S = \emptyset \quad \text{or} \quad x \in S \\ \alpha & \text{otherwise} \end{cases}$$

$$\alpha = \mathrm{max}\left[0, \max_{y \in X^i \cap S} f(y) - \min_{z \in X^i \setminus S} \left[f(z) + r\left(\sum_{j=1}^{m} \mathrm{max}\left[0, g_j(z)\right]\right)\right]\right]$$

SFP Algorithm

1. Set r, the population size N and the other genetic parameters as well as the parameters of the stopping criterion. Set i=1.
2. Generate a random initial population X^1. Set f_{best} = 1.
3. If the best individual in X^i according to $f^{\pm}$ is feasible and it gives the best fitness function value so far, update f_{best} and save that individual to x_{best}. If the stopping criterion is satisfied, stop.
4. Set X^{i+1} = ;. Carry out elitism. Repeat selection, crossover and mutation until X^{i+1} has N individuals.
5. Set i=i+1. Goto step 3.

Method of Parameter Free Penalties

- New fitness function
- Fitness of infeasible individuals does not depend on the objective function
- Infeasible solutions are directed towards the feasible region
- No parameters to be set

$$f^\circ(x) = f(x) + \textstyle\sum_{j=1}^{m} \max\left[0, g_j(x)\right] + \widehat{\theta}_i(x)$$

$$\widehat{\theta}_i(x) = \begin{cases} 0 & \text{if } x \in S \\ -f(x) & \text{if } X^i \cap S = \emptyset \\ -f(x) + \max\limits_{y \in X^i \cap S} f(y) & \text{otherwise.} \end{cases}$$

Method of Adaptive Penalties

- New fitness function
- Tries to avoid infeasible solutions by adjusting the penalty coefficient
- Parameter h – number of iterations whose best individuals are examined
- Penalty coefficient r_i is checked at each iteration i after the first h iterations. Let us denote the best individual of the iteration j by y^j. The r_i is updated according to

$$f^{\circ}(x) = f(x) + r_i\left(\sum_{j=1}^{m} \mathsf{max}\left[0, g_j(x)\right]^2 \right)$$

$$r_{i+1} = \begin{cases} \frac{r_i}{c_1} & \text{if} \quad i \geq h \;\&\; y^j \in S \;\forall\; i-h+1 \leq j \leq i \\ c_2 r_i & \text{if} \quad i \geq h \;\&\; y^j \notin S \;\forall\; i-h+1 \leq j \leq i \\ r_i & \text{otherwise} \end{cases}$$

$$c_1, c_2 > 1 \text{ and } c_1 \neq c_2$$

AP Algorithm

1. Set c_1, c_2 > 1 and r_1. Set h, the population size N and the other genetic parameters as well as the parameters of the stopping criterion. Set i=1.
2. Generate a random initial population X^1. Set f_{best} = 1.
3. Save the best individual of X^i according to $f^{\pm}$ as y^i. If this individual is feasible and it gives the best fitness function value so far, update f_{best} and save that individual to x_{best}. If the stopping criterion is satisfied, stop.
4. Set X^{i+1} = ;. Carry out elitism. Repeat selection, crossover and mutation until X^{i+1} has N individuals.
5. Calculate r_{i+1}. Set i=i+1 and goto step 3.

Comparison

- 33 test problems, 100 runs with each method
- Accuracy, efficiency (function evaluations) reliability (finding feasible solutions)
- Reliability of SFP depends on the penalty coefficient, efficiency not so good
- AP most efficient but accuracy not so good
- PFP – no parameters, always reliable, accuracy ok, efficiency not so good
- PFP better than SFP in reliability and efficiency, accuracy almost the same

Software

- Software libraries (NAG, IMSL)
- Matlab – optimization library
- Spreadsheet (Excel, Quattro Pro)
- WWW
  - Netlib
  - NEOS
  - Many other pieces of software

Applications: Flat EMFi-actuators

- Flat EMFi-actuators: complex structure with vibrating element (foil) inside acustically resistive material with special cavities
- Co-operation with VTT Automation (Tampere) and EMFiTech Ltd.
- Optimal shape design of actuators
- Optimal location of actuators in 3D
  - Based on ray tracing

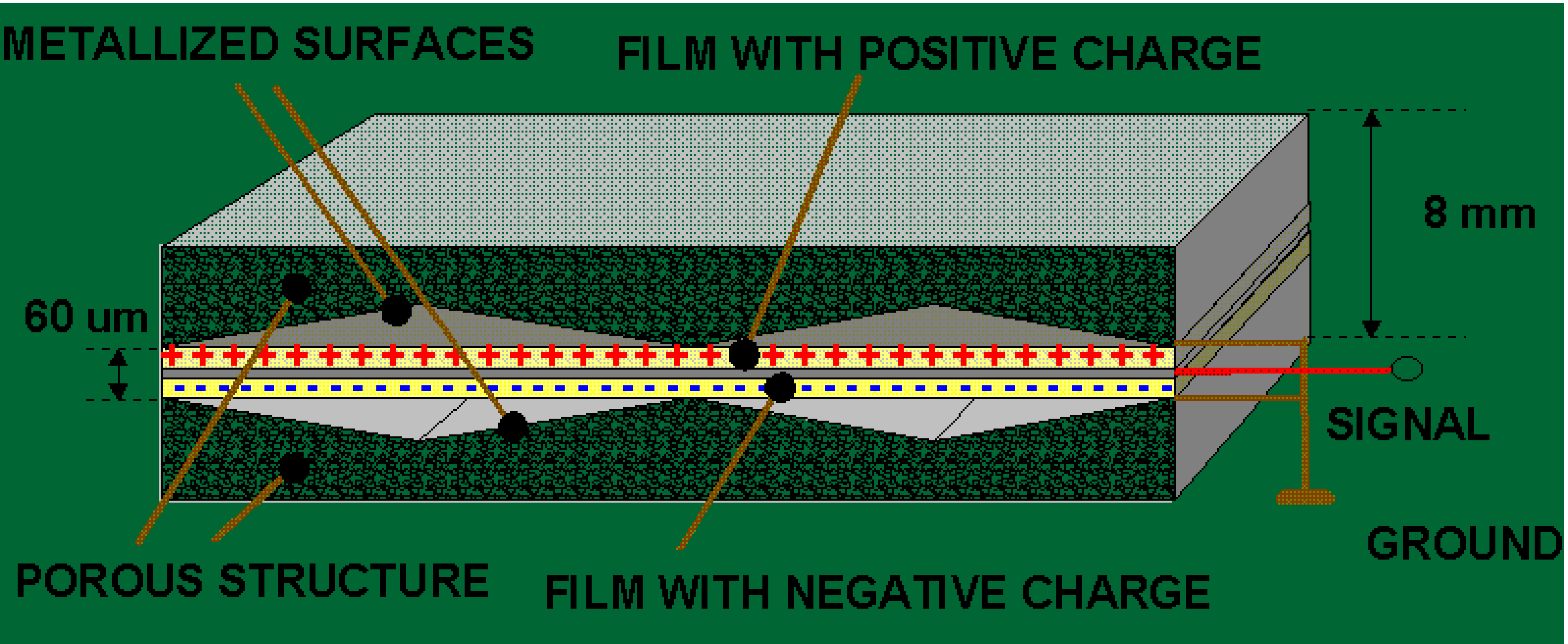
EMFi-based actuator

- Design variables:
  - geometry of the cavities
  - film tension
  - driving voltage
  - film thickness

- Objective
  - – maximize sound pressure level

  subject to

  - distortion constraint



Optimal Design of Grapple Loader

- Structural optimization
- Minimize weight
- Subject to nonlinear and nondifferentiable stress and buckling constraints

Conclusions

- No free lunch theorem
- Many, many methods exist
- Need to know when to use which kind of methods
- Trial and error is not enough
- Simulation is not enough
- Important to formulate the optimization problem properly
- Optimization can make a difference and improve performance, design, profit, safety, etc. etc.

Further Links

List available at

http://www.mit.jyu.fi/miettine/lista.html