

# A Brief Overview of Optimization Problems

Steven G. Johnson  
MIT course 18.335, Fall 2008

## Why optimization?

- In some sense, *all engineering design* is optimization: choosing **design parameters** to improve some **objective**
- Much of *data analysis* is also optimization: extracting some model parameters from data while minimizing some error measure (e.g. fitting)
- Most *business decisions* = optimization: varying some *decision parameters* to maximize profit (e.g. investment portfolios, supply chains, etc.)

## A general optimization problem

$$\min_{x \in \mathbb{R}^n} f_0(x)$$

subject to  $m$  constraints

$$f_i(x) \leq 0$$

$$i = 1, 2, \dots, m$$

minimize an **objective function**  $f_0$   
with respect to  $n$  **design parameters**  $x$   
(also called *decision parameters*, *optimization variables*, etc.)

— note that *maximizing*  $g(x)$   
corresponds to  $f_0(x) = -g(x)$

note that an *equality constraint*  
 $h(x) = 0$

yields two inequality constraints

$$f_i(x) = h(x) \text{ and } f_{i+1}(x) = -h(x)$$

(although, in practical algorithms, equality constraints  
typically require special handling)

$x$  is a **feasible point** if it  
satisfies all the constraints

**feasible region** = set of all feasible  $x$

## Important considerations

- **Global versus local** optimization
- **Convex** vs. non-convex optimization
- **Unconstrained** or **box-constrained** optimization, and other special-case constraints
- Special classes of functions (linear, etc.)
- **Differentiable** vs. non-differentiable functions
- **Gradient-based** vs. **derivative-free** algorithms
- ...
- **Zillions of different algorithms**, usually restricted to various special cases, each with strengths/weaknesses

# Global vs. Local Optimization

- For *general nonlinear* functions, *most* algorithms only guarantee a **local optimum**
  - that is, a feasible  $x_0$  such that  $f_0(x_0) \leq f_0(x)$  for all feasible  $x$  within some neighborhood  $\|x-x_0\| < R$  (for some small  $R$ )
- A *much harder* problem is to find a **global optimum**: the minimum of  $f_0$  for *all* feasible  $x$ 
  - exponentially increasing difficulty with increasing  $n$ , practically impossible to *guarantee* that you have found global minimum without knowing some special property of  $f_0$
  - many available algorithms, problem-dependent efficiencies
    - *not* just genetic algorithms or simulated annealing (which are popular, easy to implement, and thought-provoking, but usually *very slow!*)
    - for example, non-random systematic search algorithms (e.g. DIRECT), partially randomized searches (e.g. CRS2), repeated local searches from different starting points (“multistart” algorithms, e.g. MLSL), ...

## Important Convex Problems

- LP (linear programming): the objective and constraints are *affine*:  $f_i(x) = a_i^T x + \alpha_i$
- QP (quadratic programming): affine constraints + convex quadratic objective  $x^T A x + b^T x$
- SOCP (second-order cone program): LP + *cone* constraints  $\|Ax+b\|_2 \leq a^T x + \alpha$
- SDP (semidefinite programming): constraints are that  $\sum A_k x_k$  is positive-semidefinite

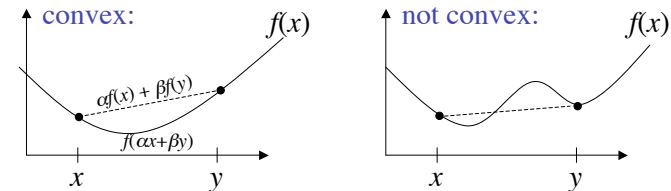
all of these have very efficient, specialized solution methods

# Convex Optimization

[ good reference: *Convex Optimization* by Boyd and Vandenberghe, free online at [www.stanford.edu/~boyd/cvxbook](http://www.stanford.edu/~boyd/cvxbook) ]

All the functions  $f_i$  ( $i=0 \dots m$ ) are **convex**:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \quad \text{where} \quad \alpha + \beta = 1, \alpha, \beta \in [0,1]$$



For a convex problem (convex objective & constraints)  
**any local optimum must be a global optimum**  
 $\Rightarrow$  efficient, robust solution methods available

## Important special constraints

- Simplest case is the **unconstrained** optimization problem:  $m=0$ 
  - e.g., line-search methods like steepest-descent, nonlinear conjugate gradients, Newton methods ...
- Next-simplest are **box constraints** (also called **bound constraints**):  $x_k^{\min} \leq x_k \leq x_k^{\max}$ 
  - easily incorporated into line-search methods and many other algorithms
  - many algorithms/software *only* handle box constraints
- ...
- Linear equality constraints  $Ax=b$ 
  - for example, can be explicitly eliminated from the problem by writing  $x=Ny+\xi$ , where  $\xi$  is a solution to  $A\xi=b$  and  $N$  is a basis for the nullspace of  $A$

## Derivatives of $f_i$

- Most-efficient algorithms typically **require user to supply the gradients  $\nabla_x f_i$**  of objective/constraints
  - you should *always* compute these analytically
    - rather than use finite-difference approximations, better to just use a derivative-free optimization algorithm
    - in principle, one can always compute  $\nabla_x f_i$  with about the same cost as  $f_i$ , using **adjoint methods**
  - gradient-based methods can find (local) optima of problems with millions of design parameters
- **Derivative-free** methods: only require  $f_i$  values
  - easier to use, can work with complicated “black-box” functions where computing gradients is inconvenient
  - *may* be only possibility for nondifferentiable problems
  - need  $> n$  function evaluations, bad for large  $n$

## Removable non-differentiability

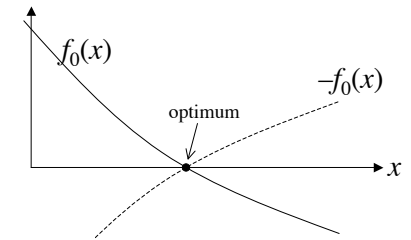
consider the *non-differentiable unconstrained* problem:

$$\min_{x \in \mathbb{R}^n} |f_0(x)|$$

equivalent to **minimax problem**:

$$\min_{x \in \mathbb{R}^n} \left( \max \{ f_0(x), -f_0(x) \} \right)$$

...still nondifferentiable...



...equivalent to **constrained** problem with a “temporary” variable  $t$ :

**differentiable!**

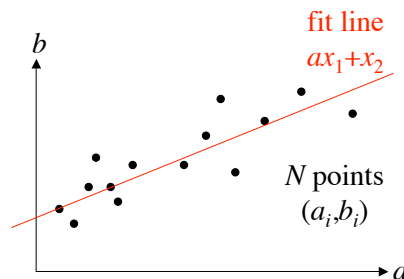
$$\min_{x \in \mathbb{R}^n, t \in \mathbb{R}} t \quad \text{subject to:} \quad \begin{array}{ll} t \geq f_0(x) & (f_1(x) = f_0(x) - t) \\ t \geq -f_0(x) & (f_2(x) = -f_0(x) - t) \end{array}$$

## Example: Chebyshev linear fitting

find the fit that minimizes the *maximum error*:

$$\min_{x_1, x_2} \left( \max_i |x_1 a_i + x_2 - b_i| \right)$$

... nondifferentiable minimax problem



equivalent to a **linear programming** problem (LP):

$$\min_{x_1, x_2, t} t$$

subject to  $2N$  constraints:

$$x_1 a_i + x_2 - b_i - t \leq 0$$

$$b_i - x_1 a_i - x_2 - t \leq 0$$

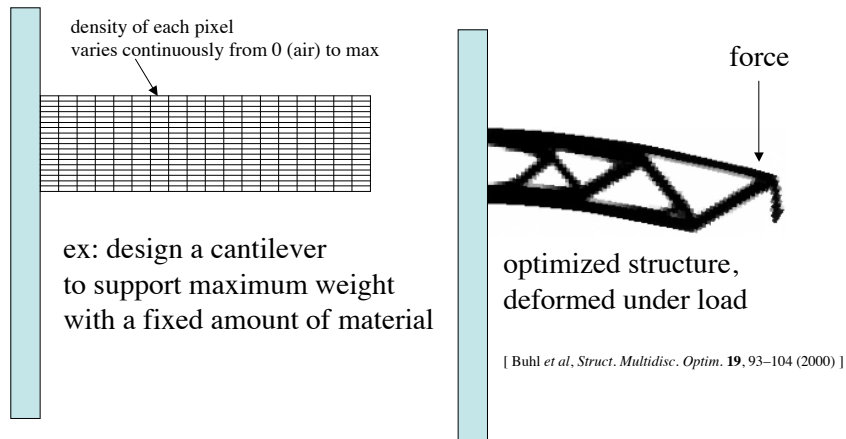
## Relaxations of Integer Programming

If  $x$  is **integer-valued** rather than real-valued (e.g.  $x \in \{0,1\}^n$ ), the resulting *integer programming* or *combinatorial optimization* problem becomes **much harder** in general.

However, useful results can often be obtained by a **continuous relaxation** of the problem — e.g., going from  $x \in \{0,1\}^n$  to  $x \in [0,1]^n$  ... at the very least, this gives an lower bound on the optimum  $f_0$

# Example: Topology Optimization

design a structure to do something, made of material A or B...  
let every *pixel* of discretized structure vary *continuously* from A to B



## Some Sources of Software

- Decision tree for optimization software:  
<http://plato.asu.edu/guide.html>  
— lists many packages for many problems
- CVX: general convex-optimization package  
<http://www.stanford.edu/~boyd/cvx>
- NLOpt: implements many nonlinear optimization algorithms  
(global/local, constrained/unconstrained, derivative/no-derivative)  
<http://ab-initio.mit.edu/nlopt>