Recent Advances in Convex Optimization

Stephen Boyd
joint work with Michael Grant, Jacob Mattingley, Yang Wang
Electrical Engineering Department, Stanford University

Stanford Computer Forum, 4/15/09
Outline

• Convex optimization
• Some (simple) examples
• Parser/solvers for convex optimization
• Real-time embedded convex optimization
Optimization

• form mathematical model of real (design, analysis, synthesis, estimation, control, . . . ) problem

• use computational algorithm to solve

• standard formulation:

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}$$

$x$ is the (decision) variable; $f$ is the objective; $C$ is the constraint set

• other formulations: multi-criterion/multi-level optimization, MDO, SAT problems, trade-off analysis, minimax, . . .
The good news

- everything\textsuperscript{1} is an optimization problem

\textsuperscript{1}i.e., much of engineering design and analysis, data analysis
The bad news

• you can’t (really) solve most optimization problems

• even simple looking problems are often intractable
Except for some special cases

- least-squares and variations (*e.g.*, optimal control, filtering)
- linear and quadratic programming
- convex optimization

well, OK, there are some other special cases
Convex optimization problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

- \(C\) is convex (closed under averaging):

\[
x, y \in C, \quad \theta \in [0, 1] \implies \theta x + (1 - \theta)y \in C
\]

- \(f\) is convex (graph of \(f\) curves upward):

\[
\theta \in [0, 1] \implies f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)
\]

- not always easy to recognize/validate convexity
Convex optimization

- (no analytical solutions, but) can solve convex optimization problems extremely well (in theory and practice)
  - get global solutions, with optimality certificate
  - problems with $10^3$–$10^5$ variables, constraints solved by generic methods on generic processor
  - (much) larger problems solved by iterative methods and/or on multiple processors
  - differentiability plays a minor role

- beautiful (and fairly complete) theory
Applications of convex optimization

• convex problems come up much more often than was once thought

• many applications recently discovered in
  – control
  – combinatorial optimization
  – signal processing
  – image processing
  – communications, networking
  – analog and digital circuit design
  – statistics, machine learning
  – finance
Some recent (general) developments

- **robust optimization methods** that handle parameter variation, optimizing average or worst-case performance, quantiles, . . .

- **$\ell_1$-based heuristics** for finding sparse solutions (compressed sensing, feature selection, . . .)

- **parser/solvers** make rapid prototyping easy

- **code generators** yield solvers that can be embedded in real-time systems
Outline

- Convex optimization
- Some (simple) examples
- Parser/solvers for convex optimization
- Real-time embedded convex optimization
Multi-period processor speed scheduling

• processor adjusts its speed $s_t \in [s^{\text{min}}, s^{\text{max}}]$ in each of $T$ time periods

• energy consumed in period $t$ is $\phi(s_t)$; total energy is $E = \sum_{t=1}^{T} \phi(s_t)$

• $n$ jobs
  – job $i$ available at $t = A_i$; must finish by deadline $t = D_i$
  – job $i$ requires total work $W_i \geq 0$

• $S_{ti} \geq 0$ is effective processor speed allocated to job $i$ in period $t$

\[
s_t = \sum_{i=1}^{n} S_{ti}, \quad \sum_{t=A_i}^{D_i} S_{ti} \geq W_i
\]
Minimum energy processor speed scheduling

- choose speeds $s_t$ and allocations $S_{ti}$ to minimize total energy $E$

$$ E = \sum_{t=1}^{T} \phi(s_t) $$

subject to

$$ s_{\text{min}} \leq s_t \leq s_{\text{max}}, \quad t = 1, \ldots, T $$

$$ s_t = \sum_{i=1}^{n} S_{ti}, \quad t = 1, \ldots, T $$

$$ \sum_{t=A_i}^{D_i} S_{ti} \geq W_i, \quad i = 1, \ldots, n $$

- a convex problem when $\phi$ is convex

- more sophisticated versions include
  - multiple processors
  - other constraints (thermal, speed slew rate, ...)
  - stochastic models for (future) data
Example

- $T = 16$ periods, $n = 12$ jobs
- $s_{\text{min}} = 1$, $s_{\text{max}} = 6$, $\phi(s_t) = s_t^2$
- jobs shown as bars over $[A_i, D_i]$ with area $\propto W_i$
Optimal and uniform schedules

- uniform schedule: \( S_{ti} = W_i / (D_i - A_i) \); gives \( E^{\text{unif}} = 374.1 \)
- optimal schedule \( S^*_{ti} \) gives \( E^* = 167.1 \)
Minimum time control with active vibration supression

- force $F(t)$ moves object modeled as 3 masses (2 vibration modes)
- tension $T(t)$ used for active vibration suppression
- goal: move object to commanded position as quickly as possible, with

\[
|F(t)| \leq 1, \quad |T(t)| \leq 0.1
\]
Ignoring vibration modes

- treat object as single mass; apply only $F$
- analytical (‘bang-bang’) solution
With vibration modes

- no analytical solution, but reduces to a quasiconvex problem
- can be solved by solving a small number of convex problems
Network utility maximization

- network with $m$ links and $n$ flows
- flow $j$ has (nonnegative) flow rate $f_j$
- each flow passes over a fixed set of links (its route)
- total link traffic (sum of flows through it) cannot exceed capacity $c_i$
- choose flow rates to maximize utility $U(f) = \sum_{i=1}^{n} U_j(f_j)$
- $U_j$ increasing and concave, e.g.,
  - $U_j(f_j) = \log f_j$ (log utility)
  - $U_j(f_j) = w_j \min\{f_j, s_j\}$ (linear with satiation)
Network utility maximization

• can express link capacity constraints as $Rf \leq c$, with

$$R_{ij} = \begin{cases} 
1 & \text{flow } j \text{ passes through link } i \\
0 & \text{otherwise}
\end{cases}$$

• NUM problem is

$$\text{maximize } U(f)$$
$$\text{subject to } Rf \leq c, \quad f \geq 0$$

a convex optimization problem

• ‘solved’ (approximately) by distributed protocols
Example

• $U_j(f_j) = \min\{f_j, s_j\}$; $c = (2, 4, 4, 2)$, $s = (2, 1, 2, 3)$
• greedy flows: optimize over $f_1$, then $f_2$, . . .

optimal, $U^* = 5$  
greedy, $U = 4$
Grasp force optimization

• choose $K$ grasping forces on object
  – resist external wrench
  – respect friction cone constraints
  – minimize maximum grasp force
• convex problem (second-order cone program):

  minimize $\max_i \| f^{(i)} \|_2$
  subject to $\sum_i Q^{(i)} f^{(i)} = f^{\text{ext}}$
  $\sum_i p^{(i)} \times (Q^{(i)} f^{(i)}) = \tau^{\text{ext}}$
  $\mu_i f_3^{(i)} \geq \left( f_1^{(i)} + f_2^{(i)} \right)^{1/2}$

  $\max$ contact force
  force equilibrium
  torque equilibrium
  friction cone constrains

variables $f^{(i)} \in \mathbb{R}^3$, $i = 1, \ldots, K$ (contact forces)
Example
Outline

• Convex optimization
• Some (simple) examples
• Parser/solvers for convex optimization
• Real-time embedded convex optimization
Parser/solvers for convex optimization

- specify convex problem in natural form
  - declare optimization variables
  - form convex objective and constraints using a specific set of atoms and calculus rules

- problem is convex-by-construction

- easy to parse, automatically transform to standard form, solve, and transform back

- implemented using object-oriented methods and/or compiler-compilers

- huge gain in productivity (rapid prototyping, teaching, research ideas)
Example (cvx)

convex problem, with variable $x \in \mathbb{R}^n$:

\[
\begin{align*}
\text{minimize} & \quad \|Ax - b\|_2 + \lambda \|x\|_1 \\
\text{subject to} & \quad Fx \leq g
\end{align*}
\]

cvx specification:

\begin{verbatim}
cvx_begin
variable x(n) % declare vector variable
minimize (norm(A*x-b,2) + lambda*norm(x,1))
subject to  F*x <= g
cvx_end
\end{verbatim}
when cvx processes this specification, it

- verifies convexity of problem
- generates equivalent IPM-compatible problem
- solves it using SDPT3 or SeDuMi
- transforms solution back to original problem

the cvx code is easy to read, understand, modify
The same example, transformed by ‘hand’

transform problem to SOCP, call SeDuMi, reconstruct solution:

```matlab
% Set up big matrices.
[m,n] = size(A); [p,n] = size(F);
AA = [speye(n), -speye(n), speye(n), sparse(n,p+m+1); ... 
    F, sparse(p,2*n), speye(p), sparse(p,m+1); ... 
    A, sparse(m,2*n+p), speye(m), sparse(m,1)];
bb = [zeros(n,1); g; b];
cc = [zeros(n,1); gamma*ones(2*n,1); zeros(m+p,1); 1];
K.f = m; K.l = 2*n+p; K.q = m + 1;  % specify cone
xx = sedumi(AA, bb, cc, K);  % solve SOCP
x = x(1:n);  % extract solution
```
Outline

• Convex optimization
• Some (simple) examples
• Parser/solvers for convex optimization
• Real-time embedded convex optimization
Real-time embedded optimization

- embed solvers in real-time applications (signal processing, control, ...)  
  *i.e.* solve an optimization problem at each time step

- requires solvers that are fast, with known maximum execution time

- used now for applications with hour/minute time-scales  
  (process control, supply chain and revenue ‘management’, trading ...)

- new methods allows millisecond/microsecond time-scales
Solving specific problems

in developing a custom solver for a specific application, we can

- exploit structure very efficiently
- determine ordering, memory allocation beforehand
- cut corners in algorithm, \( e.g., \) terminate early
- use warm start

To get \textbf{very fast} solver
Code generation

- describe optimization problem (family) in high level form
- automatically generate solver source code
- can do much at code generation time
- yields super fast solvers suitable for real-time embedded applications
Example: cvxmod specification

quadratic program, with variable $x \in \mathbb{R}^n$:

\[
\begin{align*}
\text{minimize} & \quad x^T P x + q^T x \\
\text{subject to} & \quad Gx \leq h, \quad Ax = b
\end{align*}
\]

cvxmod specification:

\[
\begin{align*}
A & = \text{matrix}(\ldots); \quad b = \text{matrix}(\ldots) \\
P & = \text{param}(‘P’, n, n, \text{psd=True}); \quad q = \text{param}(‘q’, n) \\
G & = \text{param}(‘G’, m, n); \quad h = \text{param}(‘h’, m) \\
x & = \text{optvar}(‘x’, n) \\
\text{qpfam} & = \text{problem}(\text{minimize}(\text{quadform}(x, P) + \text{tp}(q)*x), \\
& \quad [G*x \leq h, A*x = b])
\end{align*}
\]
Example: cvxmod code generation

- generate solver for problem family qpfam with
  
  \texttt{qpfam.codegen()}

- output includes \texttt{qpfam/solver.c} and ancillary files

- solve instance with
  
  \texttt{status = solve(params, vars, work);}
## Sample solve times

<table>
<thead>
<tr>
<th>problem family</th>
<th>vars</th>
<th>constrs</th>
<th>SDPT3 (ms)</th>
<th>cvxmod (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control1</td>
<td>140</td>
<td>190</td>
<td>250</td>
<td>0.4</td>
</tr>
<tr>
<td>control2</td>
<td>360</td>
<td>1080</td>
<td>1400</td>
<td>2.0</td>
</tr>
<tr>
<td>control3</td>
<td>1110</td>
<td>3180</td>
<td>3400</td>
<td>13.2</td>
</tr>
<tr>
<td>order_exec</td>
<td>20</td>
<td>41</td>
<td>490</td>
<td>0.05</td>
</tr>
<tr>
<td>net_utility</td>
<td>50</td>
<td>150</td>
<td>130</td>
<td>0.23</td>
</tr>
<tr>
<td>actuator</td>
<td>50</td>
<td>106</td>
<td>300</td>
<td>0.17</td>
</tr>
<tr>
<td>grasp</td>
<td>30</td>
<td>66</td>
<td>300</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Conclusions

- convex optimization problems come up in many application areas
- new tools make rapid prototyping easy
- new code generation methods yield solvers that can be embedded in real-time applications