Real-time convex optimization, with applications
Jacob Mattingley, Yang Wang and Argyris Zymnis
Information Systems Laboratory, Electrical Engineering, Stanford University

Convex optimization

- A convex optimization problem is of the form
  \[
  \begin{align*}
  \text{minimize} & \quad f(x) \\
  \text{subject to} & \quad x \in \mathcal{C}
  \end{align*}
  \]
  where
  - the cost function \( f \) is convex (graph of \( f \) curves upwards)
  - the constraint set \( \mathcal{C} \) is convex (closed to averaging)
- It includes linear and quadratic programming as special cases
- It can solve convex optimization problems extremely well
  - On a generic processor with a generic method for problems of up to \( 10^7 \) variables
  - With specialized iterative methods on multiple processors for larger problems
- Many applications: control, combinatorial optimization, signal processing, machine learning, finance, ...
- Recent advances in convex optimization include
  - Robust optimization methods to handle parameter variation
  - \( \ell_1 \)-based heuristics for finding sparse solutions
  - Parsers/solvers that make rapid prototyping easy
- Code generation for embedded optimization in real-time systems

Example 1: Grasp force optimization

- Choose \( K \) grasping forces on object to
  - Resist external wrench
  - Respect friction cone constraints
  - Minimize maximum grasp force
- Convex problem (second-order cone program):
  \[
  \begin{align*}
  \text{minimize} & \quad \max_i ||f(i)||_2 \\
  \text{subject to} & \quad \sum_i Q(i)f(i) = f^\text{int} \\
  & \quad \sum_i g(i) \times (Q(i)f(i)) = r^\text{int} \\
  & \quad \sum_i \mu_i f(i) \geq (f^1)^2 + (f^2)^{1/2} \\
  \end{align*}
  \]
  variables \( f(i) \in R^3, i = 1, \ldots, K \) (contact forces)
- Can solve each instance in \( 60 \mu s \)

Example 2: Minimum energy processor speed scheduling

- Processor adjusts its speed \( s_i \in [s^\text{min}, s^\text{max}] \) 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 time \( t = A_i \); must finish by deadline \( t = D_i \)
  - Job \( i \) requires total work \( W_i \geq 0 \)
- \( S_i_t \geq 0 \) is effective processor speed allocated to job \( i \) in period \( t \)
  \[
  S_i_t = \sum_{i=1}^n S_{i,t} \geq W_i
  \]
- Choose speeds \( s_i \) and allocations \( S_{i,t} \) to minimize total energy \( E \)

- When \( \phi \) is convex, can be formulated as convex problem
  \[
  \text{minimize} \quad E = \sum_{t=1}^T \phi(s_t) \\
  \text{subject to} \quad s^\text{min} \leq s_t \leq s^\text{max} \\
  \quad s_t \geq \sum_{i=1}^n S_{i,t} \geq W_i, \quad t = 1, \ldots, T \\
  \quad \sum_{t=1}^T S_{i,t} \geq W_i, \quad i = 1, \ldots, n
  \]
- Can solve each instance in \( 40 \mu s \)
- More sophisticated versions include
  - Multiple processors
  - Other constraints (thermal, speed slew rate, ...)
  - Stochastic models for (future) data

Code generation

- Say we have a quadratic program (QP), with variable \( x \in R^n \):
  \[
  \begin{align*}
  \text{minimize} & \quad x^T P x + q^T x \\
  \text{subject to} & \quad G x \leq h, \quad A x = b
  \end{align*}
  \]
- Could hand-write a fast solver, but requires much time and is complicated
- Instead, describe problem family in cvxquad (a Python package for convex optimization code generation):
  - \( A = \text{matrix}(\ldots); b = \text{matrix}(\ldots) \)
  - \( P = \text{param}(\ldots); q = \text{param}(\ldots) \)
  - \( G = \text{param}(\ldots); h = \text{param}(\ldots) \)
  - \( x = \text{optvar}(\ldots) \)
  - \( \text{gqpam} = \text{problem}(\text{minimize}(\text{quadform}(x, P) + tp(q) * x), \) \( \) \( \text{[G x <= h, A x == b])} \)
- Generate a solver for the problem family \( \text{gqpam} \) with \( \text{gqpam.codegen}() \)
- Output includes \( \text{gqpam/solver.c} \), and various ancillary files.
- Can solve an instance (in C) with
  - \( \text{status} = \text{solve}(\text{params, vars, work}) \)
- Solve times are up to \( 1000 \times \) faster than using off-the-shelf solver

\[
\phi(s) \\
\begin{array}{cc}
\text{uniform} & \text{optimal}
\end{array}
\]

\[
\begin{array}{cc}
\text{job} & \text{time}
\end{array}
\]