CVXOPT:

A Python Based Convex Optimization Suite

11 May 2012
Industrial Engineering Seminar
Andrew B. Martin

Easy and Hard

- Easy Problems - efficient and reliable solution algorithms exist

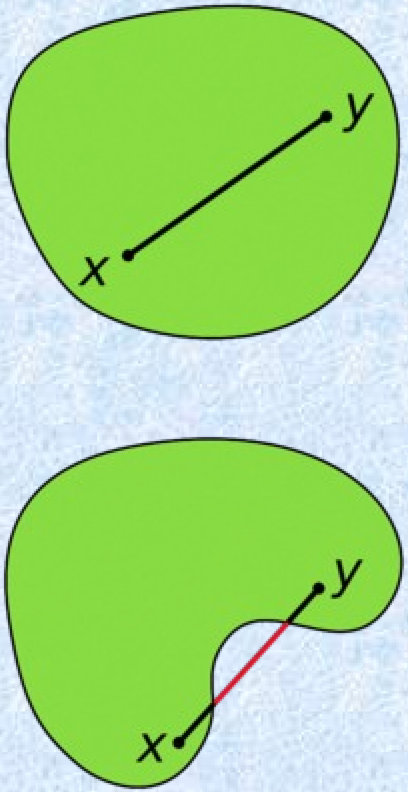
- Once distinction was between Linear/Nonlinear, now Convex/Nonconvex

Outline

- What is CVXOPT?
- How does CVXOPT solve problems?
- What are some interesting applications of CVXOPT?

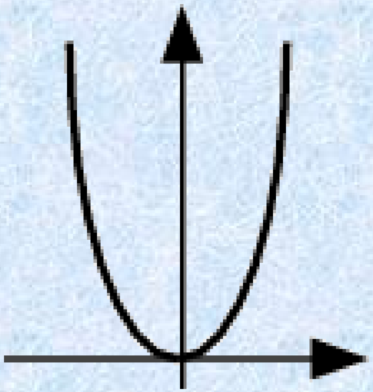
Convex Sets

- For any x,y belonging to the set C, the chord connecting x and y is contained within C.



Convex Functions

- A real valued function mapping a set X to R where X is a convex set and for any x, y belonging to X



$$f(tx+(1-t)y) \leq tf(x)+(1-t)f(y)$$

$$0 \leq t \leq 1$$

Convex Programming

$$
\begin{aligned}
&\textit{Minimize} \quad f_0(x) \\
&\textit{subject to} \quad f_i(x) \le 0 \quad i = 1,\ldots,m \\
&\qquad\qquad\quad\; a_i^T x = b_i \quad i = 1,\ldots,p
\end{aligned}
$$

- Convex objective
- Convex inequality constraint functions
- Affine equality constraint functions

Linear Programming

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Gx + s = h \\ & Ax = b \\ & s \succeq 0 \end{array} \qquad \begin{array}{ll} \text{maximize} & -h^T z - b^T y \\ \text{subject to} & G^T z + A^T y + c = 0 \\ & z \succeq 0. \end{array}$$

- Supply Chain Modeling (Forestry)

Quadratic Programming

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

- Least Squares and Constrained Least Squares
- Quantitative Finance

Second Order Cone Programming

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & G_k x + s_k = h_k, \quad k = 0, \ldots, M \\ & Ax = b \\ & s_0 \succeq 0 \\ & s_{k0} \geq \|s_{k1}\|_2, \quad k = 1, \ldots, M \end{array}$$

- Robust Linear Programming
- Truss design, robotic grasping force, antenna design

Example: Robust LP

$$\begin{aligned} &Minimize \quad c^T x \\ &subject\ to \quad a_i^T x \le b_i \quad \forall a_i \in \mathrm{E}_i \quad i = 1,...,m \end{aligned}$$

$$E_i = \{\bar{a}_i + P_i u \mid \|u\|_2 \le 1\} \quad P_i \in \Re^{nxn}$$

$$\sup\{a_i^T x \mid a_i \in \mathrm{E}_i\} \le b_i$$

$$\sup\{a_i^T x \mid a_i \in \mathrm{E}_i\} = \bar{a}_i^T x + \|P_i^T x\|_2$$

$$\begin{aligned} &Minimize \quad c^T x \\ &subject\ to \quad \bar{a}^T x + \|P_i^T x\|_2 \le b_i \quad i = 1,...,m \end{aligned}$$

Semidefinite Programming

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & G_0 x + s_0 = h_0 \\ & G_k x + \mathbf{vec}\,(s_k) = \mathbf{vec}\,(h_k), \quad k = 1, \ldots, N \\ & Ax = b \\ & s_0 \succeq 0 \\ & s_k \succeq 0, \quad k = 1, \ldots, N\end{array}$$

- Structural optimization, experiment design

Geometric Programming

$$
\begin{aligned}
&\textit{Minimize} && f_0(x) \\
&\textit{Subject to} && f_i(x) \le 1 \quad i = 1,\dots,n \\
& && h_j(x) = 1 \quad j = 1,\dots,m
\end{aligned}
$$

- Posynomial objective and inequality constraint functions.
- Monomial equality constraint functions

Globally Optimal Points

- Any locally optimal point is globally optimal!
- No concern of getting stuck at suboptimal minimums.

Overview CVXOPT

- Created by L. Vandenberghe and J. Dahl of UCLA

- Extends pythons standard libraries
  - Objects matrix and spmatrix

- Defines new modules e.g. BLAS, LAPACK, modeling and solvers

cvxopt.solvers

- Cone solvers: conelp, coneqp
- Smooth nonlinear solvers: cp, cpl
- Geometric Program solver: gp
- Customizable: kktsolver

Cone Programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx + s = h \\ & Ax = b \\ & s \succeq 0 \end{array}$$

- s belongs to C, the Cartesian product of a nonnegative orthant, a number of second order cones and a number of positive semidefinite cones

Cone Solvers

- Apply Primal-Dual Path following Interior Point algorithms with Nesterov-Todd Scaling to the previous problem

- Specify cone dimension with variable dims

- Separate QP and LP solvers

solvers.coneqp

- coneqp(P, q, G, h, A, b, dims, kktsolver)
- Linearize Central Path Equations (Newton Equations)
  - Solving these is the most expensive step
- Transform N.E. into KKT system

solvers.coneqp

Central Path Equations

$$\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix}, \quad (s, z) \succ 0, \quad z = -\mu g(s)$$

Newton Equations

$$\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -W^T W \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

solvers.coneqp

KKT System

$$\begin{bmatrix} P & A^T & G^T W^{-1} \\ A & 0 & 0 \\ W^{-T} G & 0 & -I \end{bmatrix} \begin{bmatrix} x \\ y \\ Wz \end{bmatrix} = \begin{bmatrix} a \\ b \\ W^{-T} c \end{bmatrix}$$

Constrained Regression

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x \succeq 0 \\ & \|x\|_2 \leq 1 \end{array}$$

$$A = \begin{bmatrix} 0.3 & 0.6 & -0.3 \\ -0.4 & 1.2 & 0.0 \\ -0.2 & -1.7 & 0.6 \\ -0.4 & 0.3 & -1.2 \\ 1.3 & -0.3 & -2.0 \end{bmatrix}, \qquad b = \begin{bmatrix} 1.5 \\ 0.0 \\ -1.2 \\ -0.7 \\ 0.0 \end{bmatrix}.$$

```
>> from cvxopt import matrix, solvers
>> A = matrix([ [ .3, -.4, -.2, -.4, 1.3 ], [ .6, 1.2, -1.7, .3, -.3 ], /
-.3, .0, .6, -1.2, -2.0 ] ])
>> b = matrix([ 1.5, .0, -1.2, -.7, .0])
>> m, n = A.size
>>> I = matrix(0.0, (n,n))
>> I[::n+1] = 1.0
>> G = matrix([-I, matrix(0.0, (1,n)), I])
>> h = matrix(n*[0.0] + [1.0] + n*[0.0])
>> dims = {'l': n, 'q': [n+1], 's': []}
>> x = solvers.coneqp(A.T*A, -A.T*b, G, h, dims)['x']
>> print(x)
 7.26e-01]
 6.18e-01]
 3.03e-01]
```

solvers.conelp

- Coneqp demands strict primal-dual feasibility. Conelp can detect infeasibility.
- Embeds primal and dual LPs into one LP
- Algorithm similar to coneqp except two QR factorizations used to solve KKT System

solvers.conelp

Self-dual Cone LP

$$Minimize \quad 0$$

$$\begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & C \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}, \quad (s, \kappa, z, \tau) \succ 0$$

$$
\begin{array}{ll}
\text{minimize} & -6x_1 - 4x_2 - 5x_3 \\
\text{subject to} & 16x_1 - 14x_2 + 5x_3 \leq -3 \\
& 7x_1 + 2x_2 \leq 5 \\
& \left\| \begin{bmatrix} 8x_1 + 13x_2 - 12x_3 - 2 \\ -8x_1 + 18x_2 + 6x_3 - 14 \\ x_1 - 3x_2 - 17x_3 - 13 \end{bmatrix} \right\|_2 \leq -24x_1 - 7x_2 + 15x_3 + 12 \\
& \left\| \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right\|_2 \leq 10 \\
& \begin{bmatrix} 7x_1 + 3x_2 + 9x_3 & -5x_1 + 13x_2 + 6x_3 & x_1 - 6x_2 - 6x_3 \\ -5x_1 + 13x_2 + 6x_3 & x_1 + 12x_2 - 7x_3 & -7x_1 - 10x_2 - 7x_3 \\ x_1 - 6x_2 - 6x_3 & -7x_1 - 10x_2 - 7x_3 & -4x_1 - 28x_2 - 11x_3 \end{bmatrix} \preceq \begin{bmatrix} 68 & -30 & -19 \\ -30 & 99 & 23 \\ -19 & 23 & 10 \end{bmatrix}
\end{array}
$$

dims = {'l': 2, 'q': [4, 4], 's': [3]}

Nonlinear Solvers

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_k(x) \leq 0, \quad k = 1, \ldots, m \\ & Gx \preceq h \\ & Ax = b. \end{array}$$

- Solvers.cpl for linear objective problems
- User specifies F function to evaluate gradients and hessian of constraints
- Solvers.cpl(F, G, h, A, b, kktsolver)

solvers.cpl

KKT System

$$\begin{bmatrix} H & A^T & \tilde{G}^T \\ A & 0 & 0 \\ \tilde{G} & 0 & -W^T W \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix},$$

$$H = \sum_{k=0}^{m} z_k \nabla^2 f_k(x), \quad \tilde{G} = [\nabla f_1(x) \ldots \nabla f_m(x) \; G^T]^T$$

solvers.cp

$$\begin{array}{ll}\text{minimize} & t \\ \text{subject to} & f_0(x) \leq t \\ & f_k(x) \leq 0, \quad k = 1, \ldots, m \\ & Gx \preceq h \\ & Ax = b.\end{array}$$

- For problems with a nonlinear objective function
- Problem transformed into epigraph form and solvers.cpl algorithm applied

solvers.cp

Robust Least Squares

$$Minimize \quad \sum_{k=1}^{m} \phi((Ax\text{-}b)_k)$$

$$where \quad \phi(u) = \sqrt{\rho + u^2}$$

```
from cvxopt import solvers, matrix, spdiag, sqrt, div

def robls(A, b, rho) :
    m, n = A.size
    def F(x = None, z = None) :
        if x is None : return 0, matrix(0.0, (n,1))
        y = A * x - b
        w = sqrt(rho + y **2)
        f = sum(w)
        Df = div(y, w).T * A
        if z is None : return f, Df
        H = A.T * spdiag(z[0] * rho * (w ** - 3)) * A
        return f, Df, H
    return solvers.cp(F)['x']
```

Geometric Solver

$$
\begin{array}{ll}
\text{minimize} & f_0(x) = \mathbf{lse}(F_0 x + g_0) \\
\text{subject to} & f_i(x) = \mathbf{lse}(F_i x + g_i) \leq 0, \quad i = 1, \ldots, m \\
& Gx \preceq h \\
& Ax = b
\end{array}
$$

- GP transformed to equivalent convex form and solvers.cp is applied
- Solvers.gp(F, G, h, A, b)

Exploiting Structure

- None of the previously mentioned solvers take advantage of problem structure
- User specified kktsolvers and python functions allow for customization
- Potential for better than commercial software performance

Exploiting Structure

M	N	CVXOPT	CVXOPT/BLAS	MOSEK 1.15	MOSEK 1.17
500	100	0.12	0.06	0.75	0.40
1000	100	0.22	0.11	1.53	0.81
1000	200	0.52	0.25	1.95	1.06
2000	200	1.23	0.60	3.87	2.19
1000	500	2.44	1.32	3.63	2.38
2000	500	5.00	2.68	7.44	5.11
2000	1000	17.1	9.52	32.4	12.8

- L1-norm approximation solution times for six randomly generated problems of dimension mxn (ref)

Interlude: iPython

- Interactive Programming Environment
- explore algorithms, and perform data analysis and visualization.
- It facilitates experimentation - new ideas can be tested on the fly

iPython - Features

- Magic Commands - e.g. %run
- Detailed Exception Traceback
- OS Shell Commands
- Extensive GUI support

Facility Layout Problem

- As Nonlinearly Constrained Program
- As Geometric Program
- As Quadratic Program

Nonlinearly Constrained Facility Layout Problem

$$Minimize \quad W + H$$

$$Subject\ to \quad w_k h_k \leq Area_{\min}$$

$$\mathrm{x}_{\mathrm{k}} + w_k \leq x_j$$

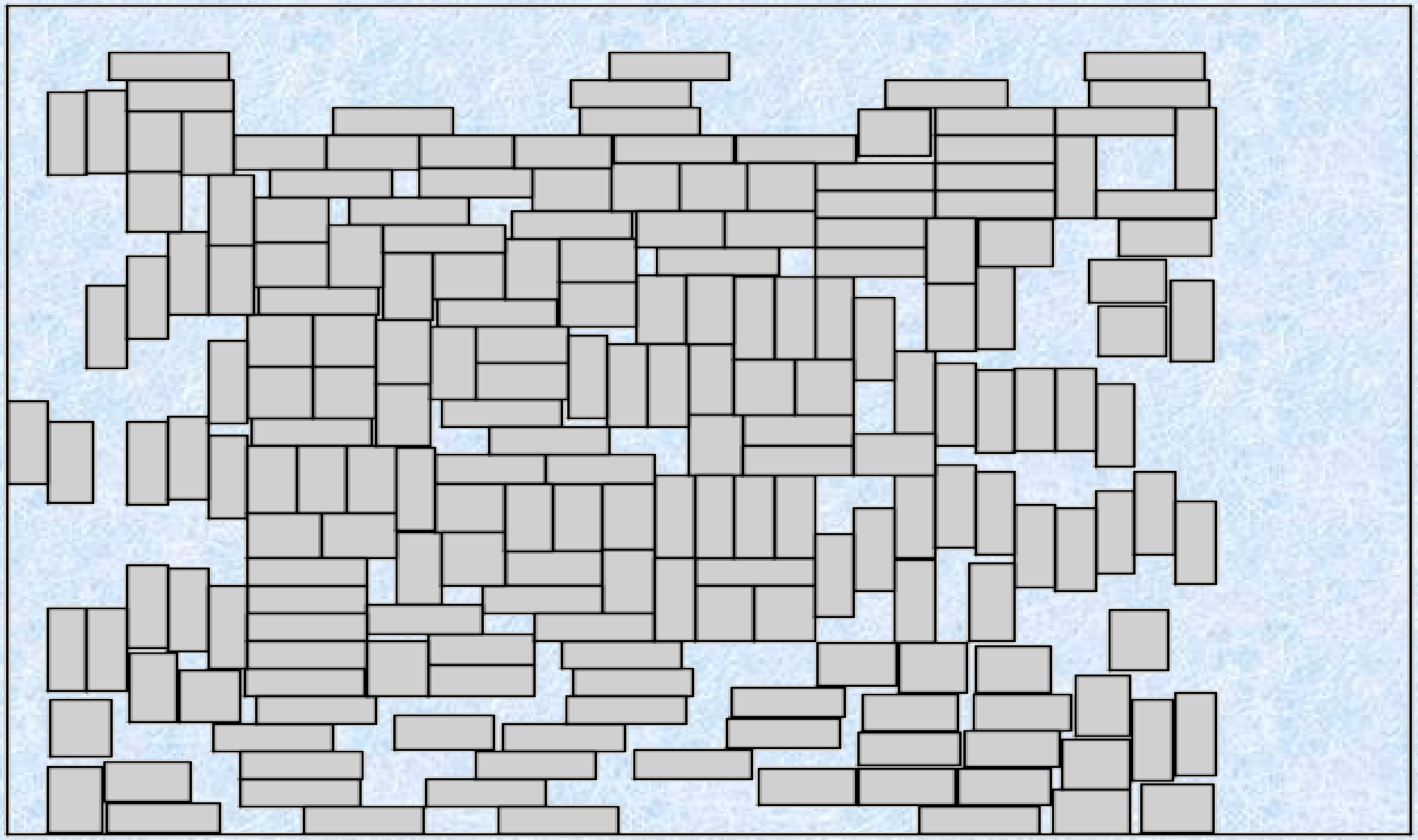
$$\mathrm{x}_{\mathrm{k}} + w_k \leq W$$

$$y_k + h_k \leq y_j$$

$$y_k + h_k \leq H$$

$$1/\alpha \leq w_k / h_k \leq \alpha$$

200 Modules



Geometric Program

$$\textit{Minimize} \quad W * H$$

$$\textit{Subject to} \quad x_j * x_i^{-1} + w_j * x_i^{-1} \le 1$$

$$x_j * W^{-1} + w_j * W^{-1} \le 1$$

$$y_j * y_i^{-1} + h_j * y_i^{-1} \le 1$$

$$y_j * H^{-1} + h_j * H^{-1} \le 1$$

$$Area_{\min} * w_j^{-1} * h_j^{-1} \le 1$$

$$1/\alpha \le w_j * h_j^{-1} \le \alpha$$

Quadratic Program

$$\text{Minimize} \quad \sum_i \sum_j flow_{i,j} * \left\| \begin{matrix} x_i - x_j \\ y_i - y_j \end{matrix} \right\|_2^2 + \beta(W + H)$$

$$\text{subject to} \quad x_j + w_j \le x_i$$

$$x_j + w_j \le W$$

$$y_j + h_j \le y_i$$

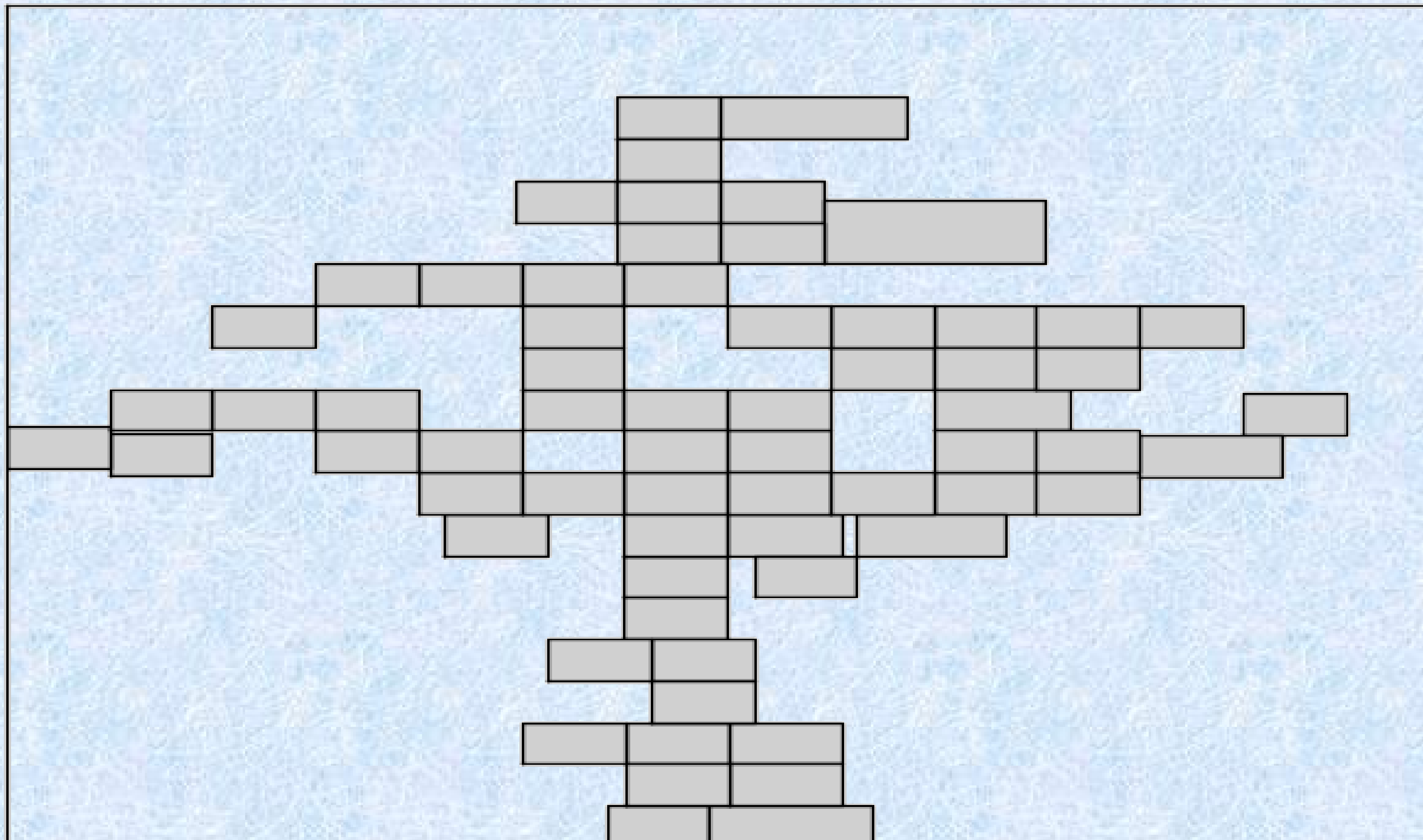
$$y_j + h_j \le H$$

$$-w_j \le -\sqrt{Area_{\min}}$$

$$-h_j \le -\sqrt{Area_{\min}}$$

$$1/\alpha \le w_j / h_j \le \alpha$$

65 Modules



Closing

- CVXOPT: Software for Convex Optimization

- How will you use it?

- andrew.b.martin@dal.ca

References