Learning for Fast and Robust Real-Time Optimization

Bartolomeo Stellato

UC Berkeley, Mechanical Engineering Seminar, November 14 2022

Autonomous systems are smarter than ever before

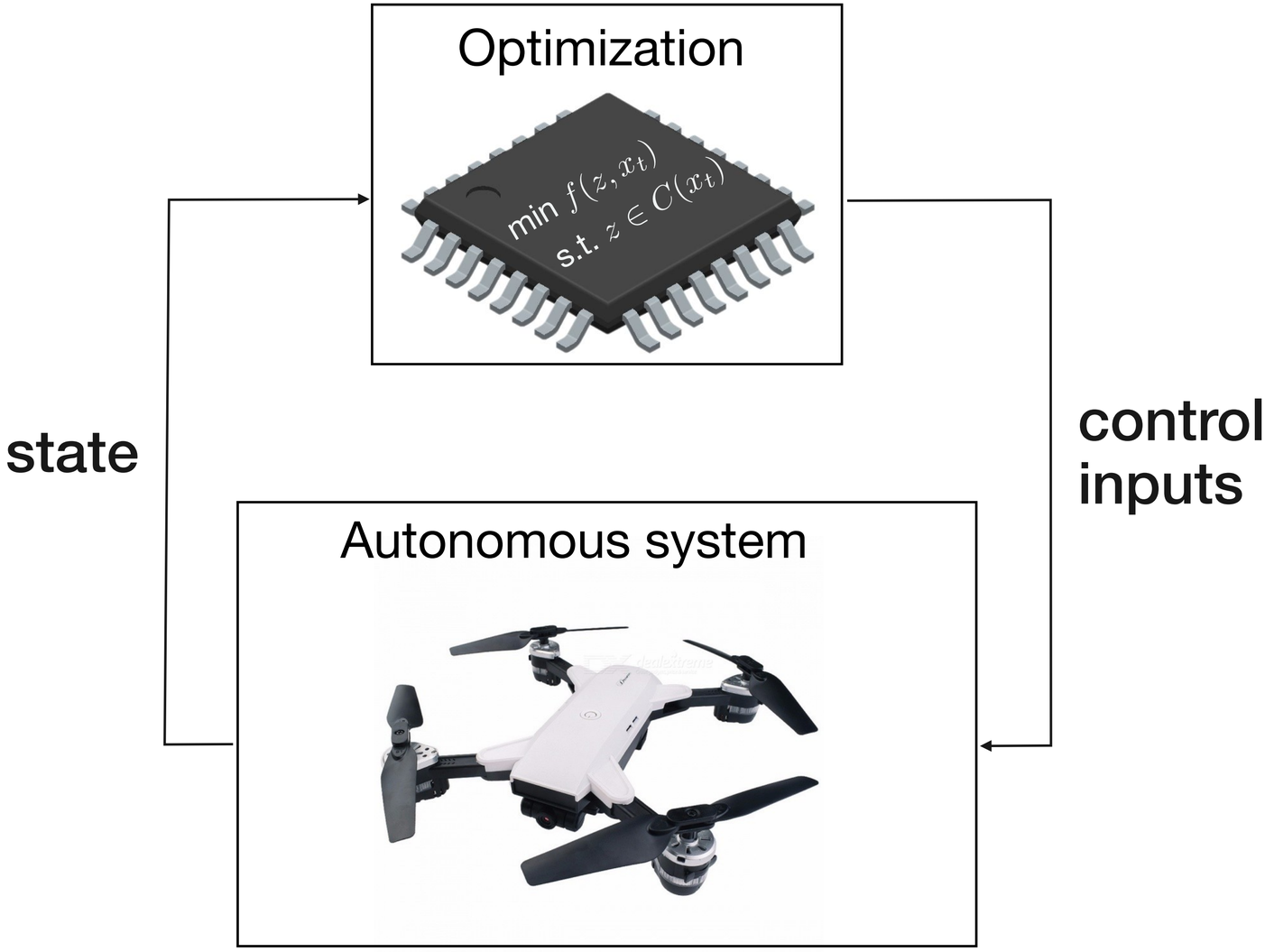
Drone racing

[Foehn, Romero, Scaramuzza]

Rocket landing

[SpaceX]

Real-Time Optimization is crucial



Key for adaptive resilient systems

- Enforce constraint satisfaction
- React to unseen scenarios

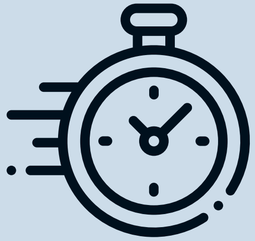
Real-world applications are still challenging!

Fast
Dynamic environments

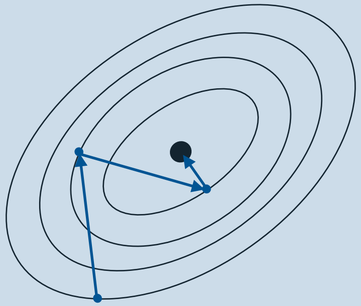
Uncertain
Extreme events

Next-generation optimization tools for Real-Time Decision Making

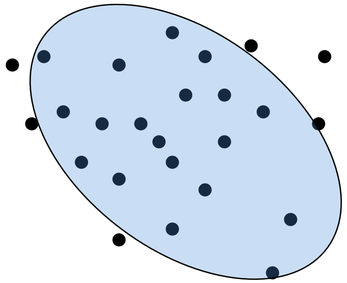
Using data to build



Fast optimization algorithms



Robust problem formulations



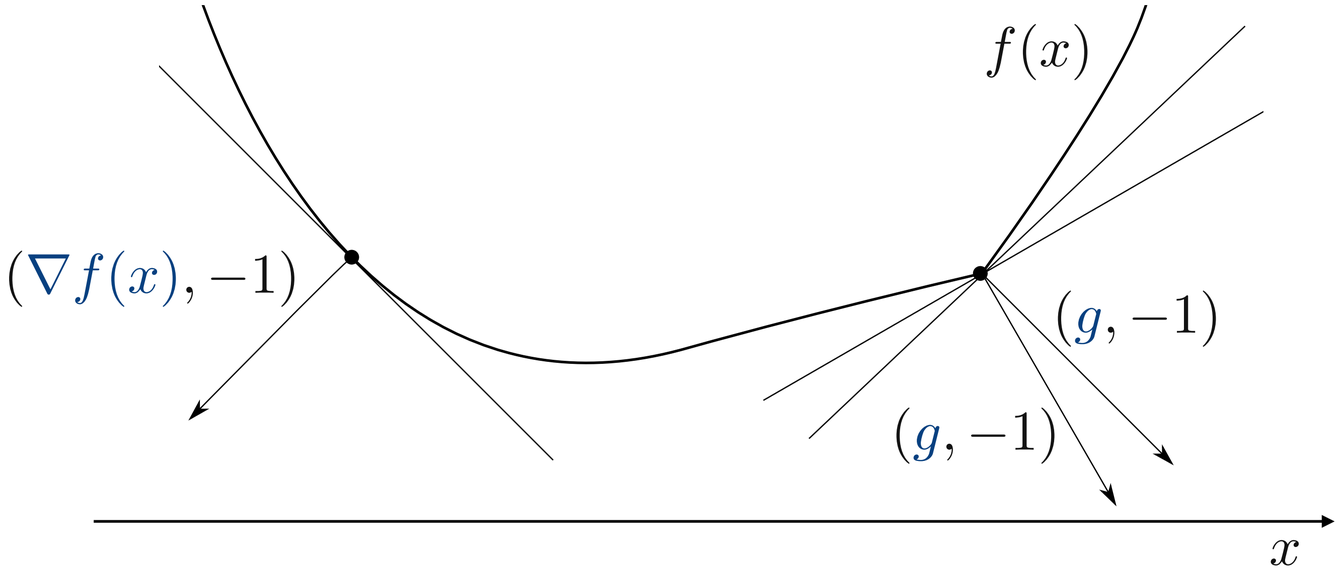
Accelerating First-Order Methods

Resurgence of first-order methods

minimize $f(x)$
with only first-order information

gradients
$\nabla f(x)$

subgradients
$g \in \partial f(x)$



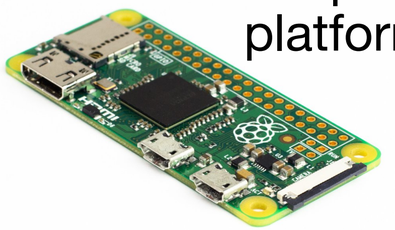
Features

- Low iteration cost
- Warm-starting

Huge scale optimization

Real-time optimization

Light computing platforms

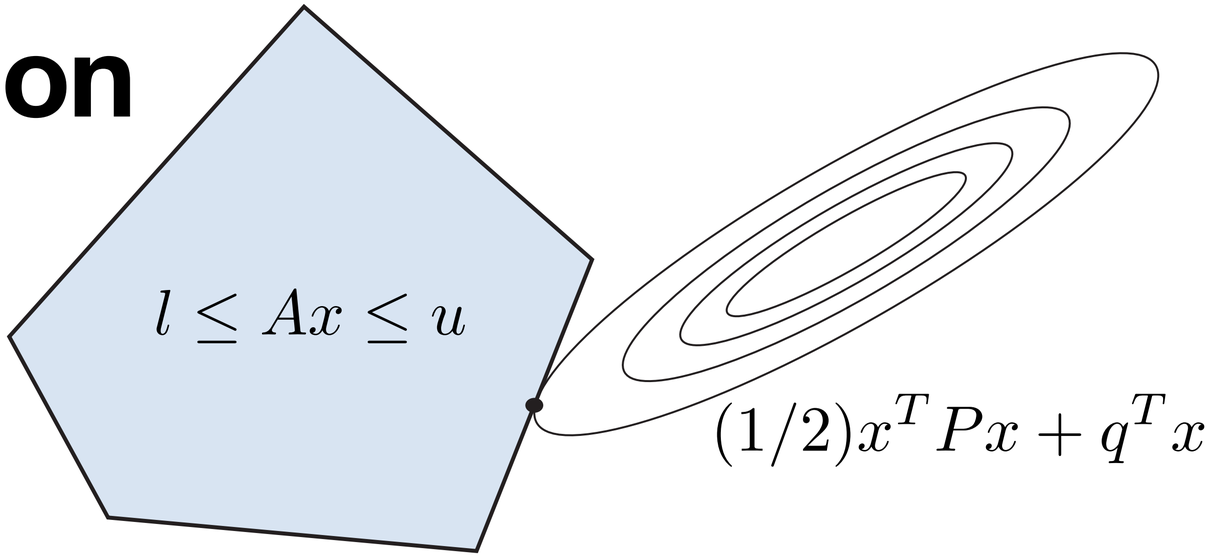


Quadratic optimization

Quadratic Optimization

minimize $(1/2)x^T Px + q^T x$

subject to $l \leq Ax \leq u$



Numerous applications

Core component in Model Predictive Control

ADMM

Alternating Direction Method of Multipliers

minimize $f(x) + g(x)$ $\longrightarrow$

Splitting

$$\begin{aligned} \text{minimize} \quad & f(\tilde{x}) + g(x) \\ \text{subject to} \quad & \tilde{x} = x \end{aligned}$$

Proximal operator

$$\mathbf{prox}_h(v) = \underset{z}{\text{argmin}} \left(h(z) + (1/2) \|z - v\|_2^2 \right)$$

Iterations

$$\begin{aligned} \tilde{x}^{k+1} &\leftarrow \mathbf{prox}_{f/\rho}(x^k - y^k/\rho) \\ x^{k+1} &\leftarrow \mathbf{prox}_{g/\rho}(\tilde{x}^{k+1} + y^k/\rho) \\ y^{k+1} &\leftarrow y^k + \rho \left(\tilde{x}^{k+1} - x^{k+1} \right) \end{aligned}$$

How do we split the QP?

$$
\begin{array}{ll}
\text{minimize} & (1/2)x^T P x + q^T x \\
\text{subject to} & Ax = z \\
 & z \in \mathcal{C}
\end{array}
$$

f (for $(1/2)x^T P x + q^T x$ and $Ax = z$)

g (for $z \in \mathcal{C}$)

Splitting formulation

$f(\tilde{x}, \tilde{z})$ and $g(x, z)$

$$
\begin{array}{ll}
\text{minimize} & (1/2)\tilde{x}^T P \tilde{x} + q^T \tilde{x} + \mathcal{I}_{Ax=z}(\tilde{x}, \tilde{z}) + \mathcal{I}_{\mathcal{C}}(z) \\
\text{subject to} & \tilde{x} = x \\
 & \tilde{z} = z
\end{array}
$$

Diagonal step sizes

$\tilde{x} = x$ → σ

$\tilde{z} = z$ → ρ

OSQP Algorithm

Problem

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & l \leq Ax \leq u \end{array}$$

Algorithm in a nutshell

Linear system solve

Easy operations

$$x^{k+1} \leftarrow \text{Solve} \quad (P + \sigma I + \rho A^T A)x = \sigma x^k - q + A^T(\rho z^k - y^k)$$

$$z^{k+1} \leftarrow \Pi(Ax^{k+1} + \rho^{-1} y^k)$$

$$y^{k+1} \leftarrow y^k + \rho(Ax^{k+1} - z^{k+1})$$

always solvable!

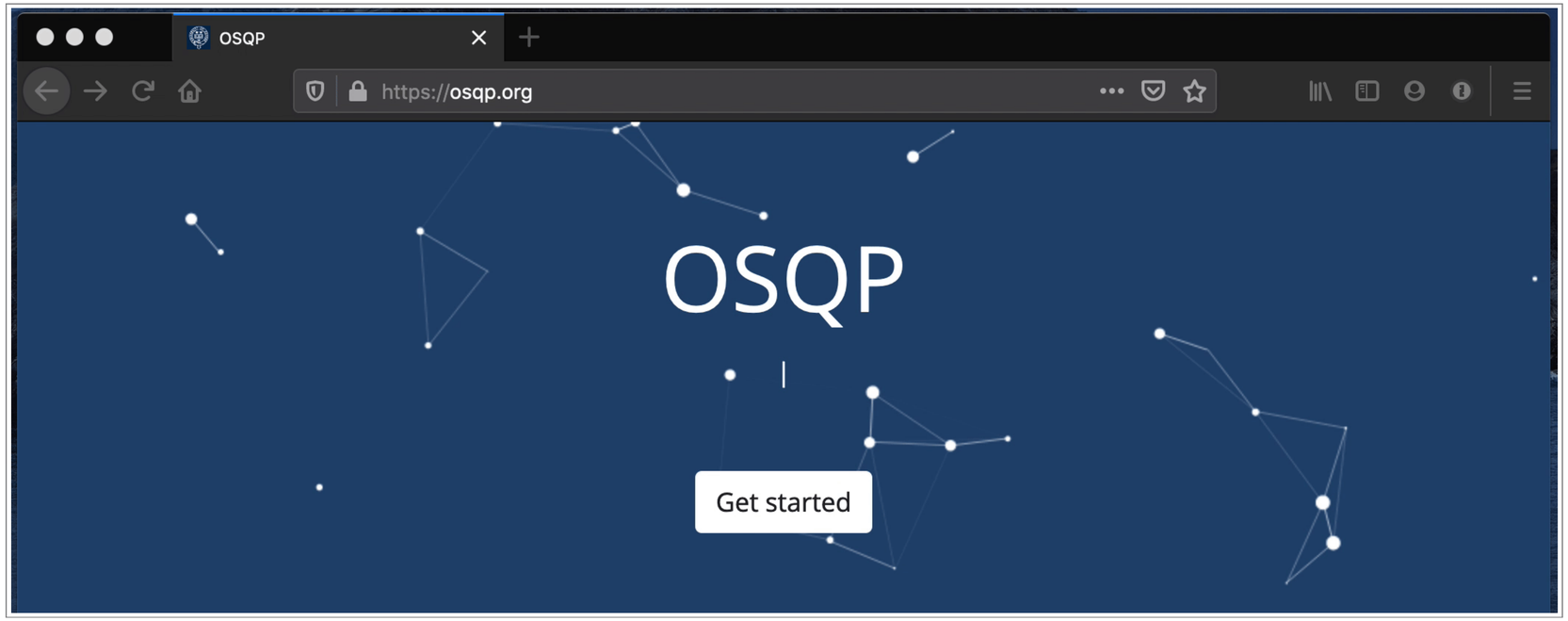
Efficient

Robust

Detects Infeasibility

OSQP

Operator Splitting solver for Quadratic Programs



OSQP: An Operator Splitting Solver for Quadratic Programs
B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd
Mathematical Programming Computation, 2020

Mathematical Programming Computation
Best Paper Award

Users

Academia










Industry







30 million downloads!

[https://pepy.tech/project/osqp]

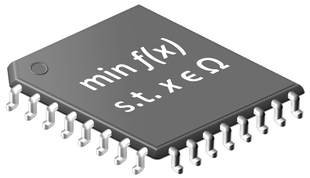
Code generation with OSQP

```python
# Create OSQP object
m = osqp.OSQP()

# Initialize solver
m.setup(P, q, A, l, u,
        settings)

# Generate C code
m.codegen('folder_name')
```

Optimized C code

```c
// Main ADMM algorithm
for (iter = 1; iter <= work->settings->max_iter; iter ++) {
    // Update x_prev, z_prev (preallocated, no malloc)
    swap_vectors(&(work->x), &(work->x_prev));
    swap_vectors(&(work->z), &(work->z_prev));

    /* ADMM STEPS */
    /* Compute \tilde{x}^{k+1}, \tilde{z}^{k+1} */
    update_xz_tilde(work);

    /* Compute x^{k+1} */
    update_x(work);

    /* Compute z^{k+1} */
    update_z(work);

    /* Compute y^{k+1} */
    update_y(work);

    /* End of ADMM Steps */

    #ifdef CTRLC
    // Check the interrupt signal
    if (isInterrupted()) {
        update_status(work->info, OSQP_SIGINT);
        c_print("Solver interrupted\n");
        endInterruptListener();
        return 1;   // exitflag
    }
    #endif
```

Embedded Hardware



New in OSQP 1.0: Code generation from C to C

Python API calls C code generation

```python
# Create an OSQP object
prob = osqp.OSQP()

# Setup workspace and change alpha parameter
prob.setup(P, q, A, l, u, alpha=1.0)

# Generate C code
prob.codegen(
    'folder',                   # Output folder for auto-generated code
    prefix='mysolver_',         # Prefix for filenames and C variables

    parameters='vectors',       # What do we wish to update in the generated code?
                                # 'vectors'/'matrices'
    use_float=False,            # Use single precision in generated code?
    printing_enable=False,      # Enable solver printing?
    profiling_enable=False,     # Enable solver profiling?
    interrupt_enable=False,     # Enable user interrupt (Ctrl-C)?
    include_codegen_src=True,   # Include headers/sources/Makefile in the output folder,
                                # creating a self-contained compilable folder?

    compile=False,              # Compile the python wrapper?
    python_ext_name='pyosqp',   # Name of the generated python extension
)
```

Naming

Solver options

Codegen wrapper

Code generation from C to C

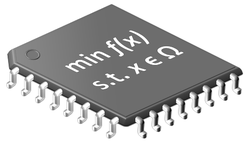
This is what gets called…

```c
exitflag = osqp_setup(&solver, P, q, A, l, u, m, n, settings);

/* Test codegen */
OSQPCodegenDefines *defs = (OSQPCodegenDefines *)malloc(sizeof(OSQPCodegenDefines));

defs->float_type = 0;         /* Use doubles */
defs->printing_enable = 0;    /* Don't enable printing */
defs->profiling_enable = 0;   /* Don't enable profiling */
defs->interrupt_enable = 0;   /* Don't enable interrupts */

/* Generate code that allows only vector updates */
defs->embedded_mode = 1;
osqp_codegen(solver, vecDirPath, "vec_prefix_", defs);

/* Generate code that allows both vector and matrix updates */
defs->embedded_mode = 2;
osqp_codegen(solver, matDirPath, "mat_prefix", defs);
```

Desktop C solver

Embeddable code

- No dynamic memory allocation
- Division-free



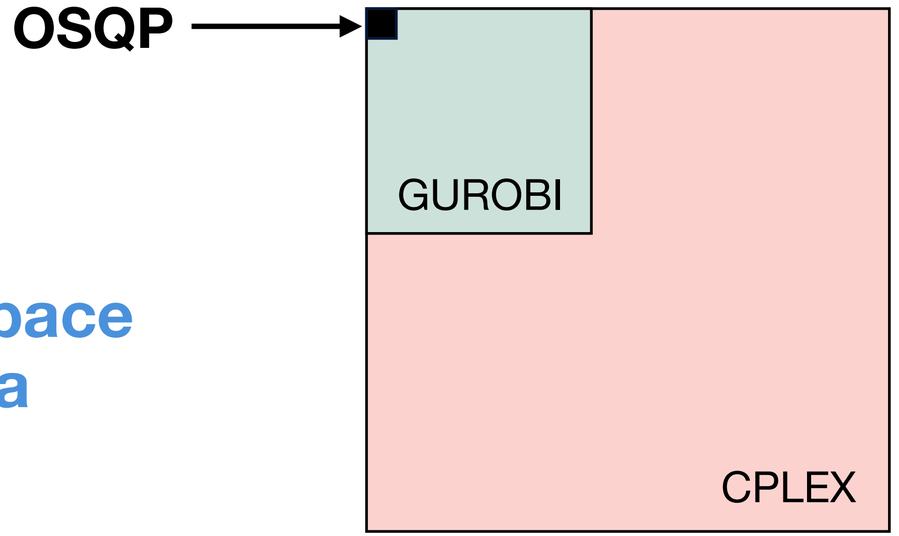
Code generation results

Self-contained and simplified directory structure

```
$ tree out
out
├── emosqp.c
├── inc
│   ├── private
│   │   ├── algebra_impl.h
⋮   ⋮   ⋮
│   │   └── version.h
│   └── public
│       ├── csc_type.h
│       ├── osqp_api_constants.h
│       ├── osqp_api_functions.h
│       ├── osqp_api_types.h
│       ├── osqp_api_utils.h
│       ├── osqp_export_define.h
│       └── osqp.h
├── Makefile
├── osqp_configure.h
├── mysolver_workspace.c
├── mysolver_workspace.h
└── src
    ├── algebra_libs.c
    ⋮
    ├── osqp_api.c
    ⋮
    └── vector.c
```

Workspace data

Compiled code size ~80kb (low footprint)

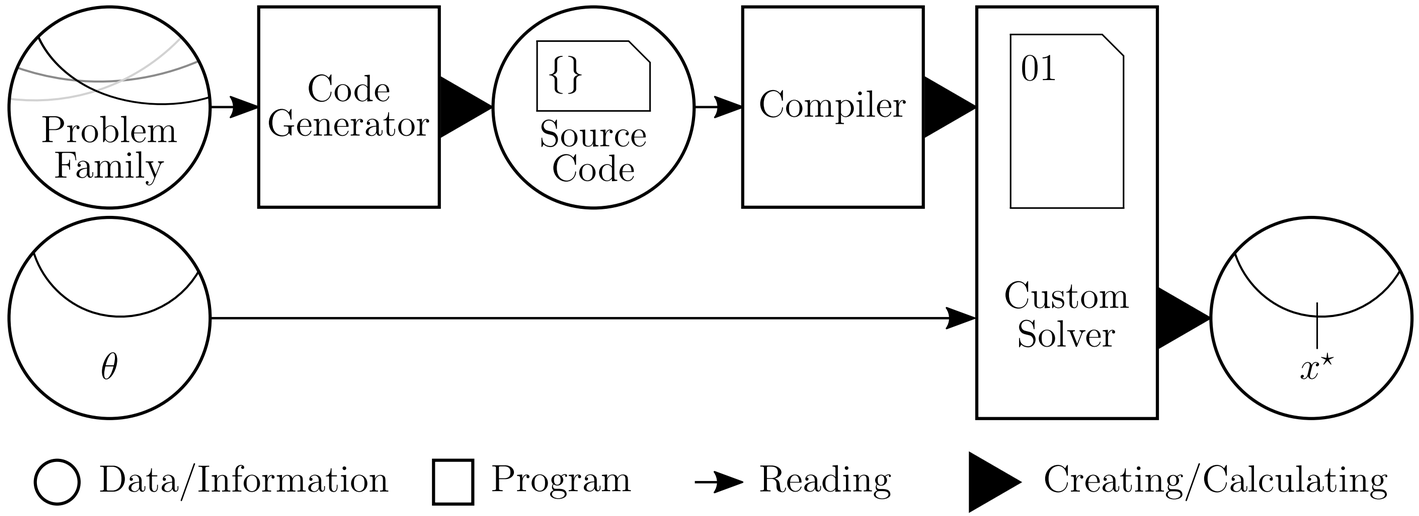


300x Reduction!

Code generation for parametric convex optimization

OSQP is integrated in CVXPYgen

https://pypi.org/project/cvxpygen/



Example

$$\begin{array}{ll} \text{minimize} & \|Gx - h\|^2 \\ \text{subject to} & x \geq 0 \end{array}$$

$$\theta = (G, h)$$

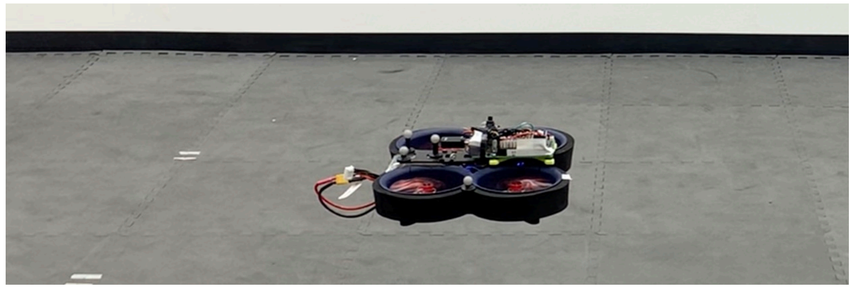
```python
import cvxpy as cp
from cvxpygen import cpg

# model problem
x = cp.Variable(n, name='x')
G = cp.Parameter((m,n), name='G')
h = cp.Parameter(m, name='h')
p = cp.Problem(cp.Minimize(cp.sum_squares(G@x-h)),
               [x>=0])
# generate code
cpg.generate_code(p)
```

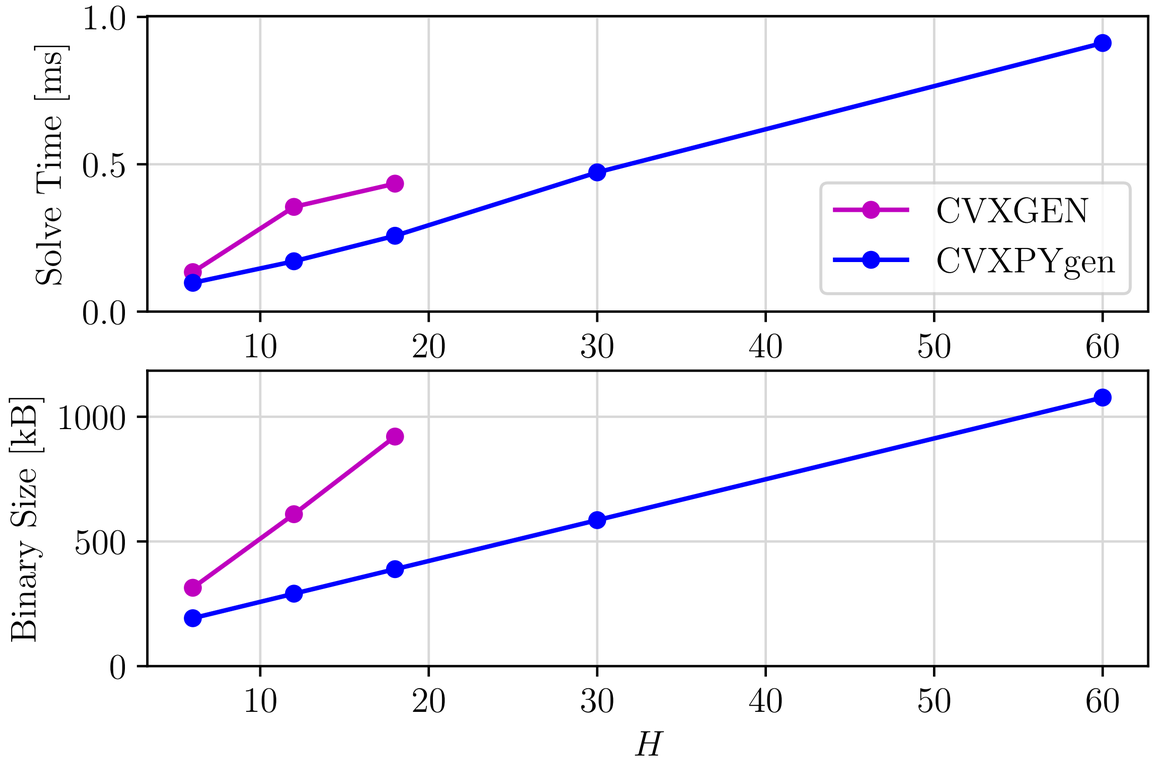
Embedded Code Generation with CVXPY
Schaller, Banjac, Diamond, Agrawal, Stellato, Boyd
IEEE Control Systems Letters, 2022

CVXPYgen to deploy OSQP on a quadcopter

MPC for quadcopter control

Deployed on
Intel Atom x5-Z8350





Embedded Code Generation with CVXPY
Schaller, Banjac, Diamond, Agrawal, Stellato, Boyd
IEEE Control Systems Letters, 2022

Back to the OSQP Algorithm

Problem

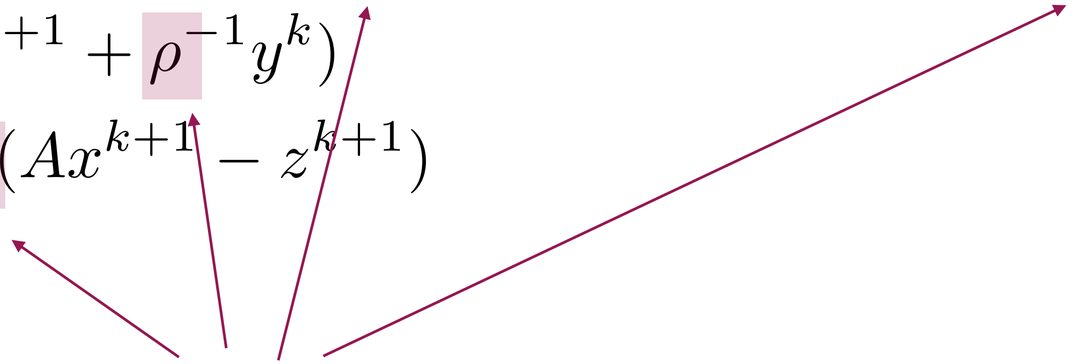
$$\begin{array}{ll} \text{minimize} & (1/2)x^T Px + q^T x \\ \text{subject to} & l \leq Ax \leq u \end{array}$$

Algorithm in a nutshell

$$x^{k+1} \leftarrow \text{Solve} \quad (P + \sigma I + \rho A^T A)x = \sigma x^k - q + A^T(\rho z^k - y^k)$$

$$z^{k+1} \leftarrow \Pi(Ax^{k+1} + \rho^{-1} y^k)$$

$$y^{k+1} \leftarrow y^k + \rho(Ax^{k+1} - z^{k+1})$$



Value of ρ can significantly change convergence behavior

Critical for real-time optimization!

How do we obtain fast convergence?

minimize $(1/2)x^T Px + q^T x$

subject to $Ax = z$

$l \leq z \leq u$

Primal residual

$r^k_{\text{prim}} = Ax^k - z^k$

Dual residual

$r^k_{\text{dual}} = Px^k + q + A^T y^k$

Intuition

(linear system solution with $\sigma = 0$)

$$\left(P + \rho A^T A\right) x^{k+1} = -q + A^T(\rho z^k - y^k)$$

$\rho = \infty$ $\qquad$ $\rho = 0$

small primal residual $\quad A^T A x^{k+1} = A^T z^k$

$Px^{k+1} = -q - A^T y^k \quad$ small dual residual

What's the optimal ρ?

Constraint-wise step size

minimize $(1/2)x^T Px + q^T x$

subject to $l \leq Ax \leq u$

Tight constraints

$l_i = (Ax^\star)_i$ or $(Ax^\star)_i = u_i$

Diagonal step size

$\rho = (\rho_1, \ldots, \rho_m)$

Never tight

$l_i = -\infty$ and $u_i = \infty$

↓

$\rho_i = 0$

Otherwise

↓

Balance residuals

$\rho_i^{k+1} \leftarrow \rho_i^k \sqrt{\|r_{\text{prim}}\| / \|r_{\text{dual}}\|}$

Always tight

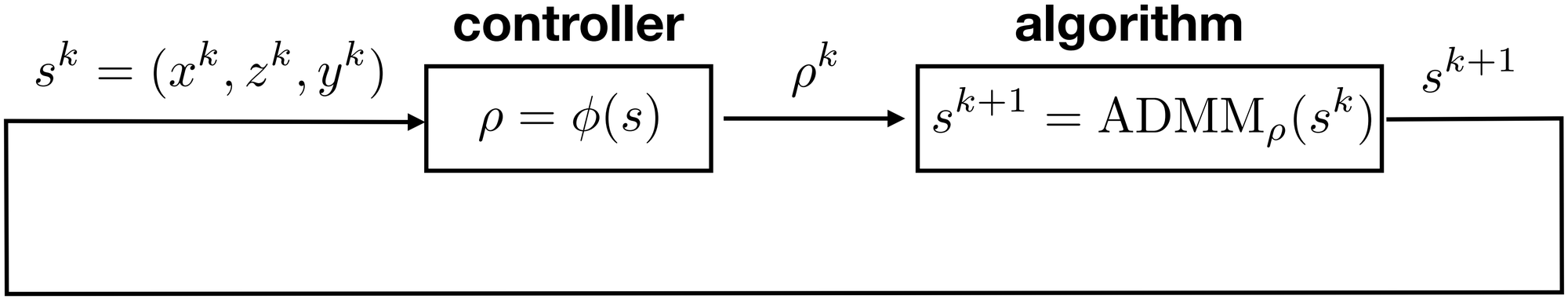
$l_i = u_i \neq \infty$

↓

$\rho_i = \infty$

Can we learn a better update rule from data?

Step size choice as a control problem



Stage cost

$$\ell(s) = \begin{cases} 1 & \text{if not converged} \\ 0 & \text{if converged} \end{cases}$$

Cumulative cost

$$J = \mathbf{E} \sum_{k=1}^{\infty} \gamma^k \ell(s^k)$$

Train with
Deep Policy Gradient methods (TD3)

Constraint-wise control policy

$$\phi(s) = \begin{bmatrix} \phi_c(s_1) \\ \phi_c(s_2) \\ \vdots \\ \phi_c(s_m) \end{bmatrix}$$

Per-constraint update rule

$$\rho_i = \phi_c(s_i)$$

Per-constraint state

$$s_i = \begin{bmatrix} \min(z_i - l_i, u_i - z_i) \\ (Ax)_i - z_i \\ y_i \end{bmatrix}$$

slacks

infeasibility

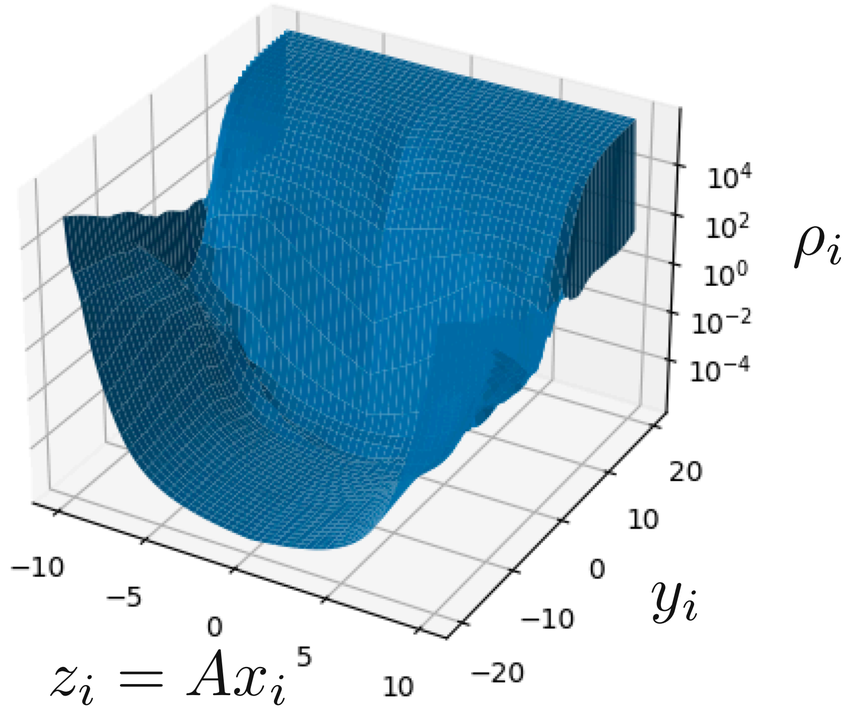
dual variable

Generalize to different dimensions

Low-dimensional state per constraint

Small NN policy $\phi_c(s_i)$

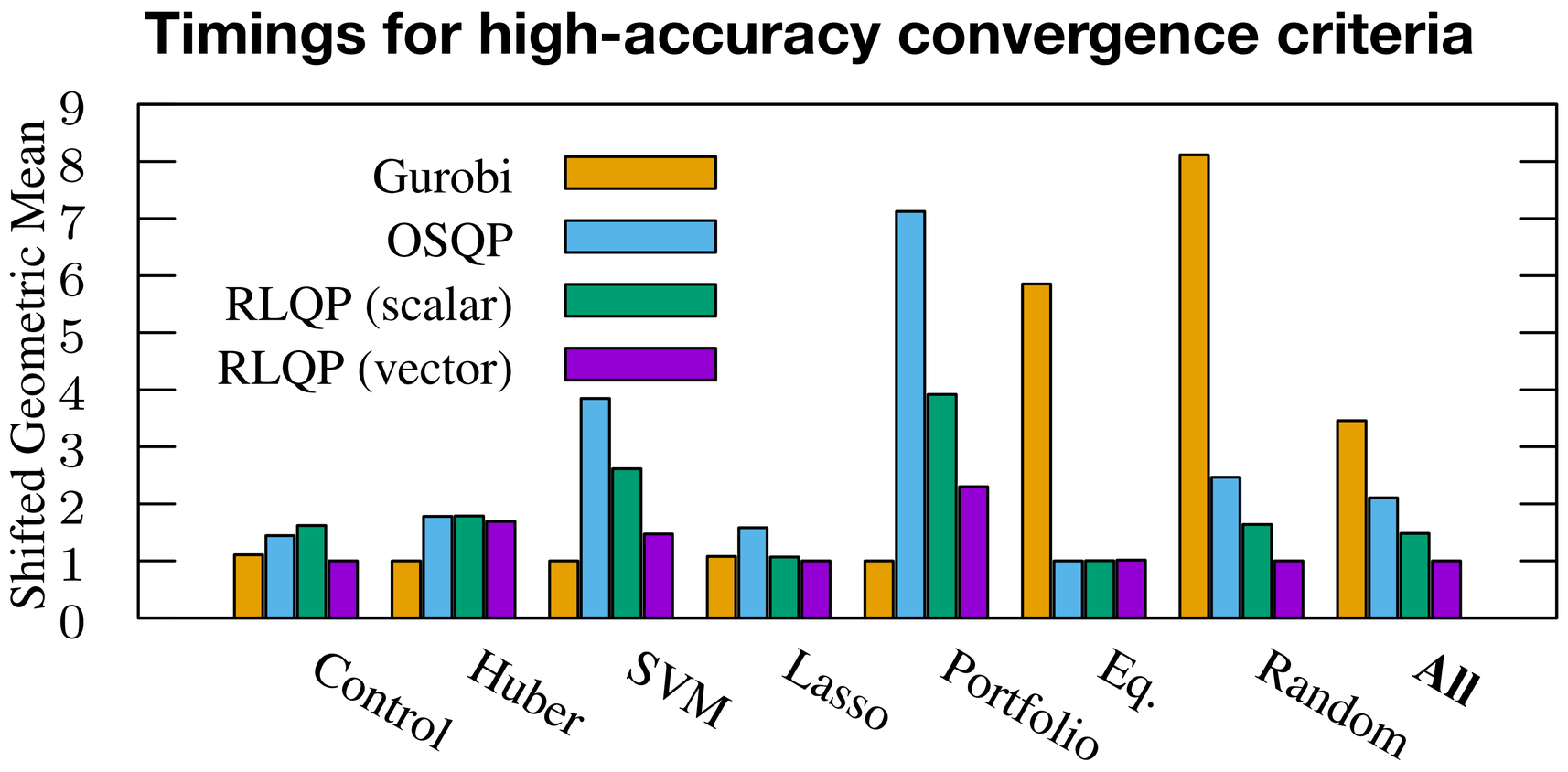
Visualize learned policy



Interpretable policy

High step size ρ_i
when we reach bounds
$z_i \approx l_i$ and $z_i \approx u_i$.

Performance with step size learning



Up to 3x faster than Gurobi

Accelerating Quadratic Optimization with Reinforcement Learning
J. Ichnowski, P. Jain, B. Stellato, G. Banjac, M. Luo, F. Borrelli, J. Gonzalez, I. Stoica, and K. Goldberg. *NeurIPS*, 2021

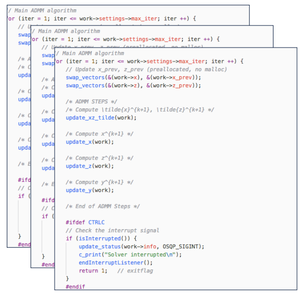
OSQP features coming soon to 1.0

Feeling impatient! Try it in branch `develop-1.0`

`</>` github.com/osqp/{osqp,osqp-python,OSQP.jl}

Improved embedded code generation

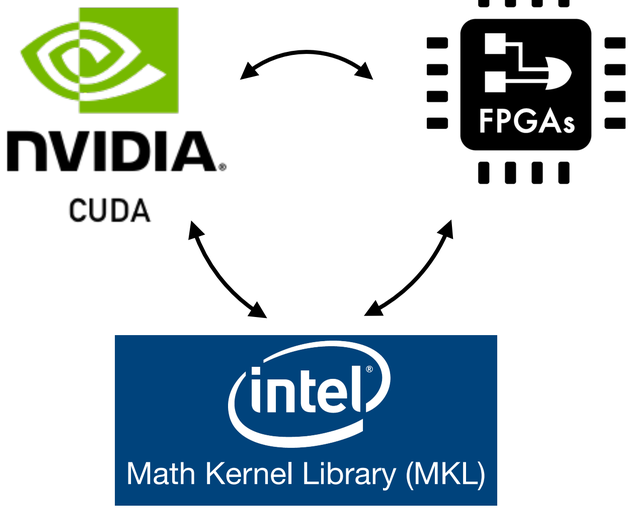
```
# Create OSQP object
m = osqp.OSQP()

# Initialize solver
m.setup(P, q, A, l, u,
        settings)

# Generate C code
m.codegen('folder_name')
```



Embedded Hardware

Code generation from C to C

Modular linear algebra



Differentiable layers

PyTorch

Ongoing and future research

Algorithm certification

Parametric optimization

$$\begin{array}{ll} \text{minimize} & f(x,\theta) \\ \text{subject to} & g(x,\theta) \le 0 \end{array}$$

parameters

Solve with first-order method

$$x^{k+1} = T_\theta(x^k)$$

Can we guarantee convergence?

$$\theta \sim \mathbf{P}$$

Algorithm certification

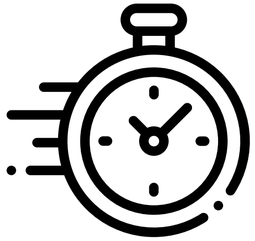
$$\begin{array}{ll} \text{maximize} & \mathbf{E}_\theta \|x^K - T_\theta(x^K)\| \\ \text{subject to} & x^{k+1} = T_\theta(x^k), \quad k = 0, \ldots, K-1 \\ & x^0 = X(\theta) \end{array}$$

Final fixed-point residual

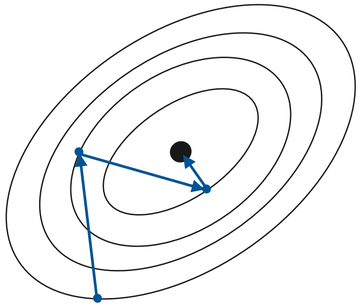
Initial iterate set

Next-generation optimization tools for Real-Time Decision Making

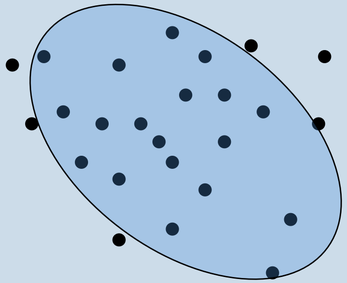
Using data to build



Fast optimization algorithms



Robust problem formulations



Decision-making under Uncertainty

It is hard to make decisions under uncertainty

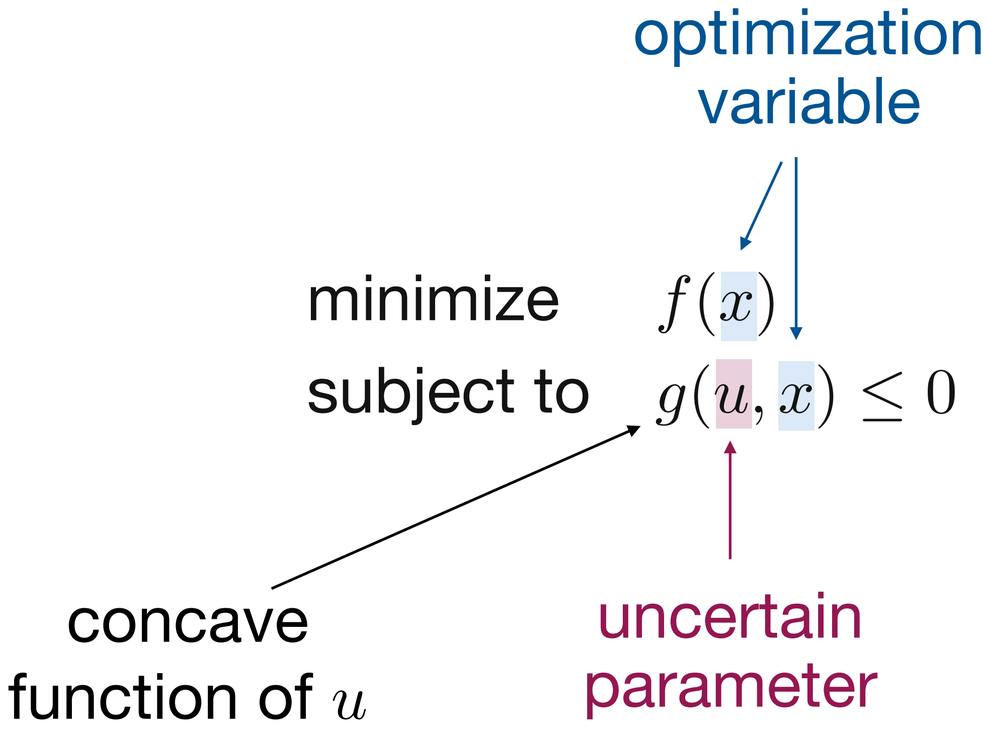
Transportation

Finance

Energy

But we have data!

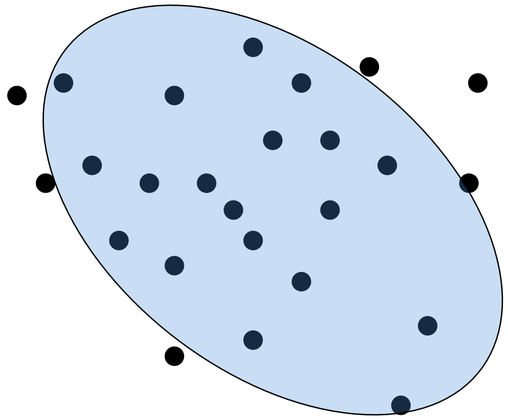
Problem setup with uncertain constraints



We want to guarantee constraint satisfaction

How do we solve it?

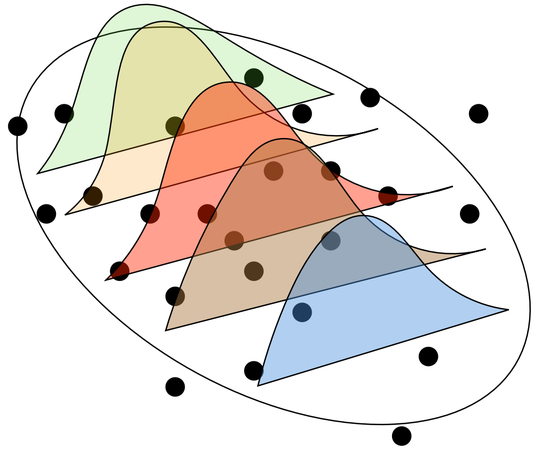
Robust optimization (RO)



✓ Tractable, expressive

❌ Can be conservative

Distributionally Robust Optimization (DRO)



✓ Can be less conservative

❌ Computationally expensive

Can we get the best of both worlds?

Probabilistic guarantees

$$\mathbf{E}(g(u,x)) \leq 0$$

$u \sim P$
(but we never know P!)

Data
$\mathcal{D} = \{d_i\}_{i=1}^{N}$

Data-driven probabilistic guarantees

Product Distribution → $\mathbf{P}^N\left(\mathbf{E}(g(u,\hat{x}_N)) \leq 0\right) \geq 1-\beta$ ← probability of constraint satisfaction

data-driven solution → $\hat{x}_N$

Mean Robust Optimization (MRO)

Data

$\mathcal{D}_N$

Machine Learning Clustering

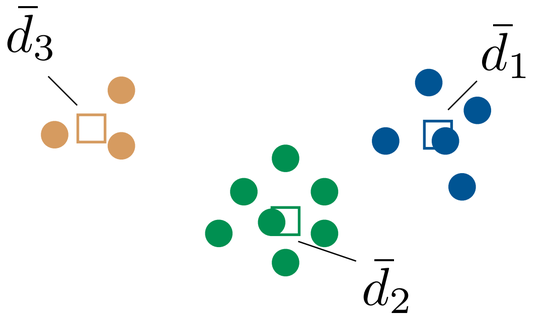
Uncertainty Set

$\mathcal{U}(K, \epsilon)$

Mean Robust Optimization Problem

Data-driven solution

$\hat{x}_N$

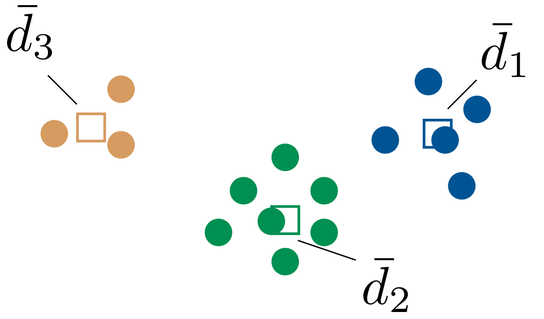


$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & \bar{g}(u, x) \leq 0 \quad \forall u \in \mathcal{U}(K, \epsilon) \end{aligned}$$

constraint function

uncertainty set

Uncertainty set



cluster weights

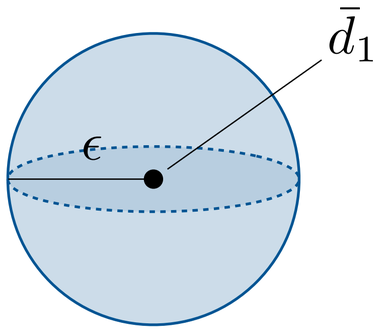
order

$$\mathcal{U}(K,\epsilon) = \left\{ u = (v_1, \ldots, v_K) \;\middle|\; \sum_{k=1}^{K} w_k \|v_k - \bar{d}_k\|^p \leq \epsilon^p \right\}$$

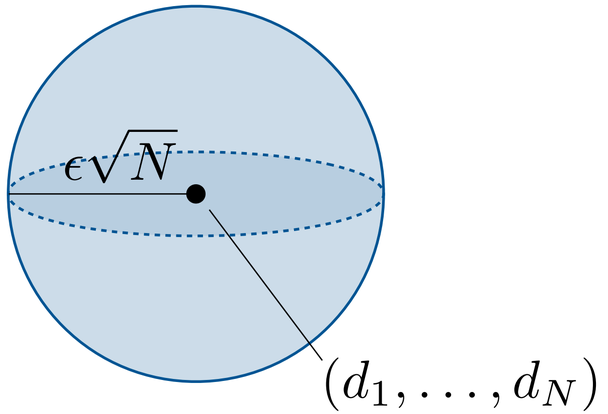
cluster centers

Examples

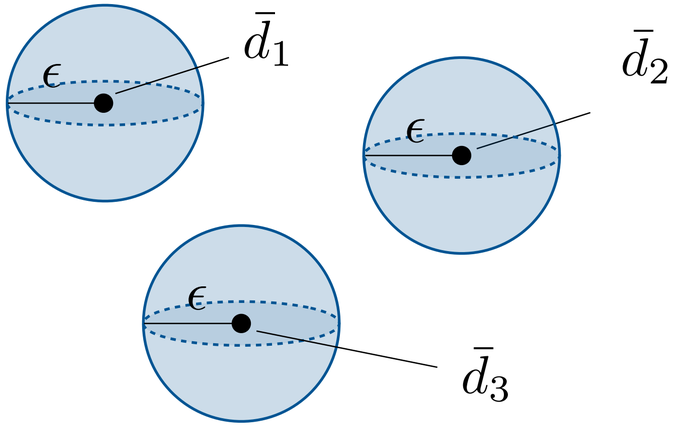
$K = 1$



$K = N, p = 2$



$K = 3, p = \infty$



Mean Robust Optimization Problem

uncertainty set

$$\left\{ \sum_{k=1}^{K} w_k \|v_k - \bar{d}_k\|^p \leq \epsilon^p \right\}$$

Uncertain variable lifting

$$u = (v_1, \ldots, v_K)$$

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & \bar{g}(u, x) \leq 0 \quad \forall u \in \mathcal{U}(K, \epsilon) \end{aligned}$$

constraint function

$$\sum_{k=1}^{K} w_k g(v_k, x)$$

Solving the MRO problem

Dualize constraint $\bar{g}(u, x) \leq 0, \ \forall u \in \mathcal{U}(K, \epsilon)$

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & \sum_{k=1}^{K} w_k s_k \leq 0 \\
& [-g]^*(z_k, x) - z_k^T \bar{d}_k + \phi(p)\lambda \|z_k/\lambda\|_*^{p/(p-1)} + \lambda \epsilon^p \leq s_k, \quad k = 1, \ldots, K \\
& \lambda \geq 0
\end{array}
$$

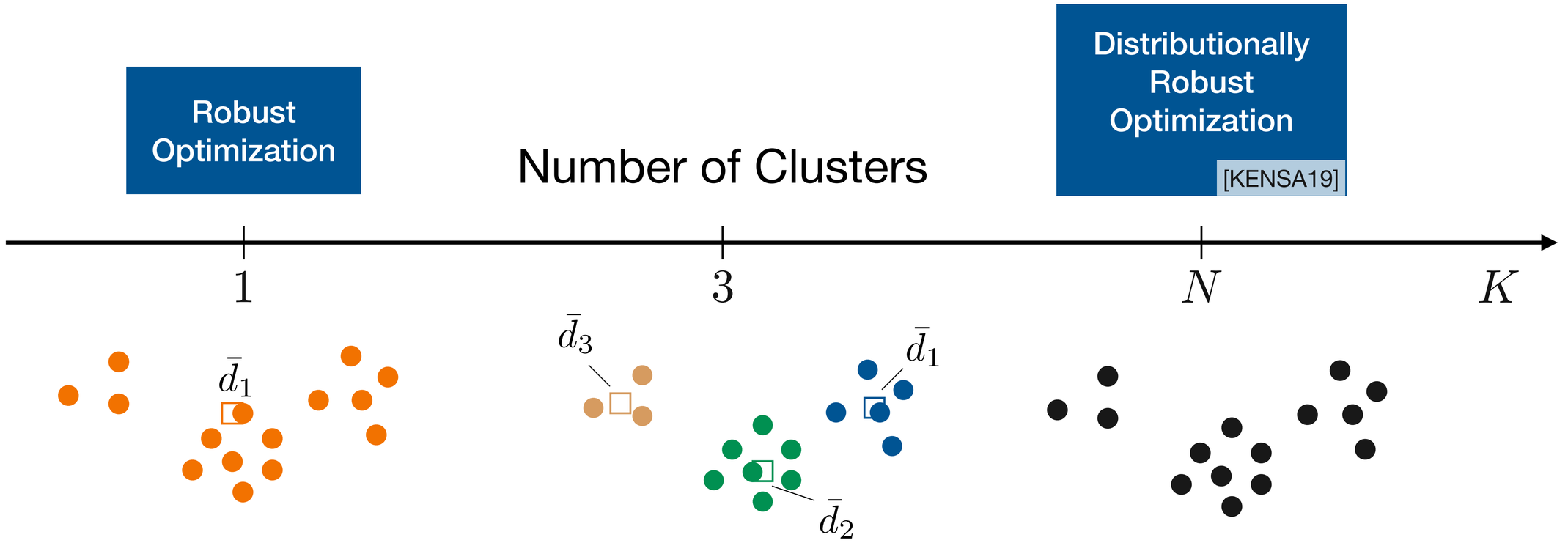
conjugate function

cluster centers

function of $p \geq 1$
$\phi(p) \to 1$ as $p \to \infty$
$\phi(1) = 0$

It can be very expensive when K is large (e.g., $K = N$)

MRO bridges between RO and DRO



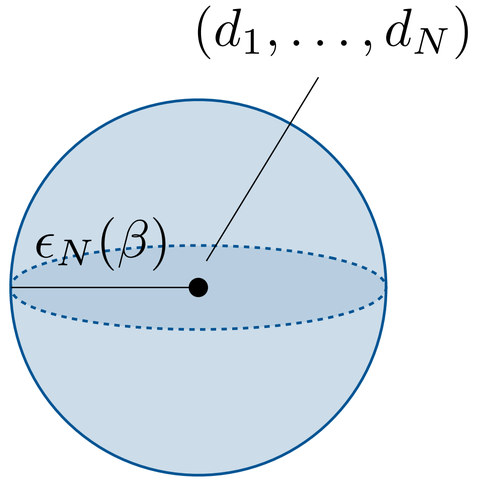
[KENSA19] D Kuhn, P M Esfahani, V A Nguyen, and S Shafieezadeh-Abadeh, “Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning

Satisfying the probabilistic guarantees

probability of constraint satisfaction

uncertainty set radius

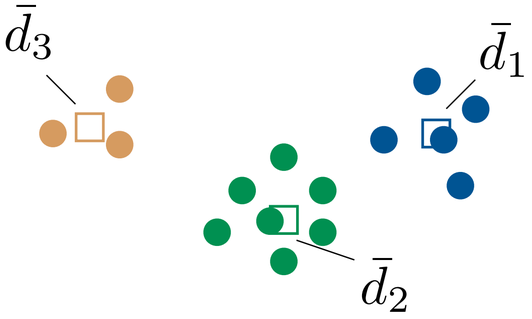
$$\mathbf{P}^N\left(\mathbf{E}(g(u,\hat{x}_N)) \leq 0\right) \geq 1-\beta \quad \xrightarrow{\text{light-tailed}} \quad \mathcal{U}(N, \epsilon_N(\beta))$$



MRO clustering

$$\mathcal{U}(K, \epsilon_N(\beta) + \eta_N(K))$$

$$\max_{i \in C_k} \|d_i - \bar{d}_k\|$$



Quite conservative bounds… can we do better?

Bounding the conservatism

MRO constraint

$$\bar{g}(u, x) \leq 0 \quad \forall u \in \mathcal{U}(K, \epsilon)$$

Worst-case values

$$\bar{g}^N(x) = \underset{u \in \mathcal{U}(N,\epsilon)}{\text{maximize}} \quad \bar{g}(u, x)$$

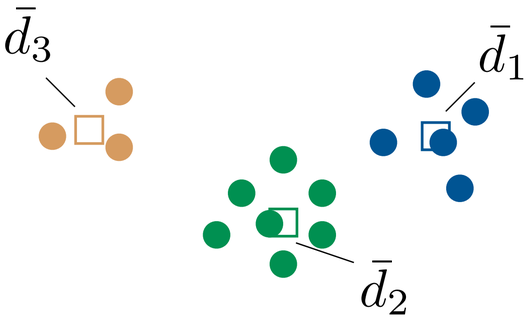
$$\bar{g}^K(x) = \underset{u \in \mathcal{U}(K,\epsilon)}{\text{maximize}} \quad \bar{g}(u, x)$$

Theorem

If $-g$ is L-smooth in u, we have

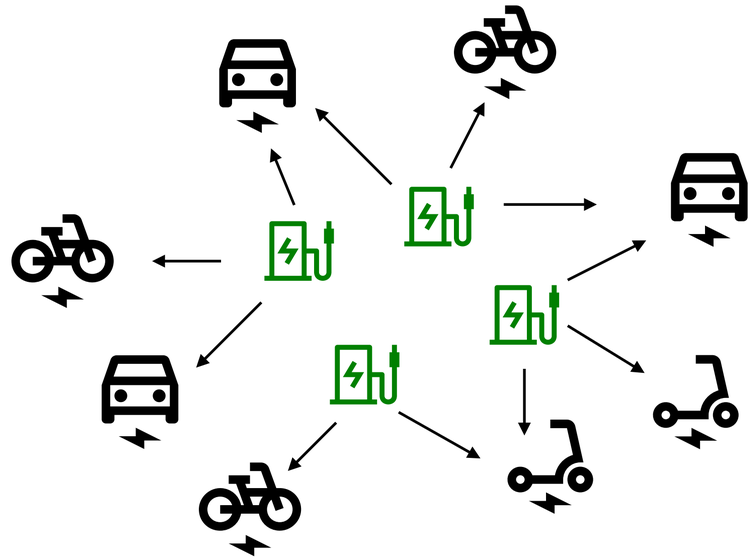
$$\bar{g}^N(x) \leq \bar{g}^K(x) \leq \bar{g}^N(x) + \frac{L}{2} D(K)$$

$D(K)$ ⟵ min clustering objective $\frac{1}{N} \sum_{k=1}^{K} \sum_{d_i \in C_k} \|d_i - \bar{d}_k\|^2$



When g is affine in u ($L = 0$), clustering makes no difference to the optimal value or optimal solution

Facility location



cost of opening charging stations

cost of energy distribution

$$
\begin{aligned}
\text{minimize} \quad & c^T x + \mathbf{tr}(C^T X) \\
\text{subject to} \quad & \mathbf{1}^T X_j = 1, \quad j = 1, \ldots, m \\
& (X^T)_i u \leq r_i x_i, \quad i = 1, \ldots, n \\
& x \in \{0,1\}^n, \quad X \in \mathbf{R}^{n \times m}
\end{aligned}
$$

capacity constraints

vector of uncertain energy demands

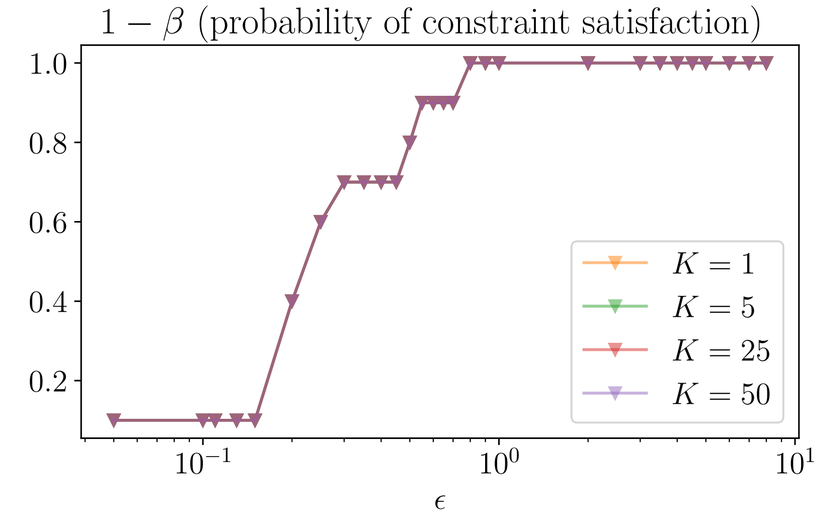
$$g(u, x, X)_i = (X^T)_i u - r_i x_i$$

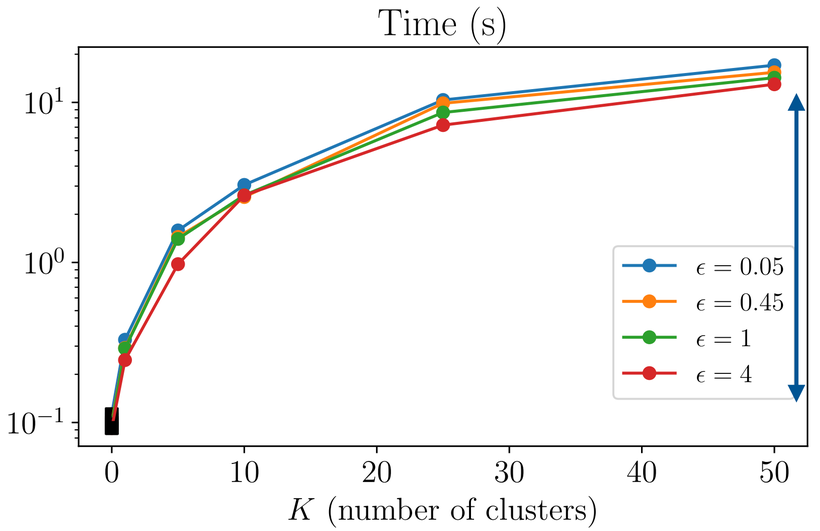
MRO formulation

$$
\begin{aligned}
\text{minimize} \quad & c^T x + \mathbf{tr}(C^T X) \\
\text{subject to} \quad & \mathbf{1}^T X_j = 1, \quad j = 1, \ldots, m \\
& \bar{g}(u, x, X)_i \leq 0, \quad \forall u \in \mathcal{U}(K, \epsilon), \quad i = 1, \ldots, n \\
& x \in \{0,1\}^n, \quad X \in \mathbf{R}^{n \times m}
\end{aligned}
$$

Facility location results

$n = 5$ locations, $m = 120$ users, $N = 50$ samples





clustering does not affect constraint satisfaction

100x speedups!

Capital budgeting example

total net present value (NPV)

Problem

maximize $\eta(u)^T x$

subject to $a^T x \leq b$ ← budget constraint

$x \in \{0,1\}^n$

$$g(u,x,\tau) = -\eta(u)^T x - \tau$$

MRO formulation

minimize τ

subject to $\bar{g}(u,x,\tau) \leq 0, \quad \forall u \in \mathcal{U}(K,\epsilon)$

$a^T x \leq b$

$x \in \{0,1\}^n$

NPV of project j

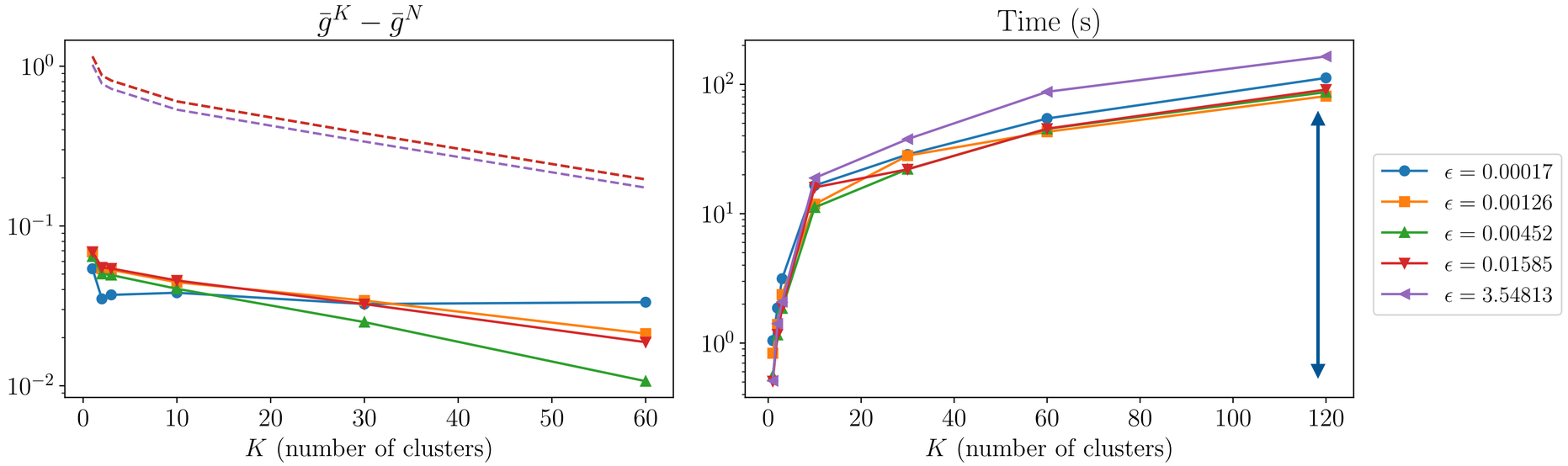
$$\eta_j(u) = \sum_{t=1}^{T} \frac{F_{jt}}{(1+u_j)^t}$$

F_{jt} ← cash flow

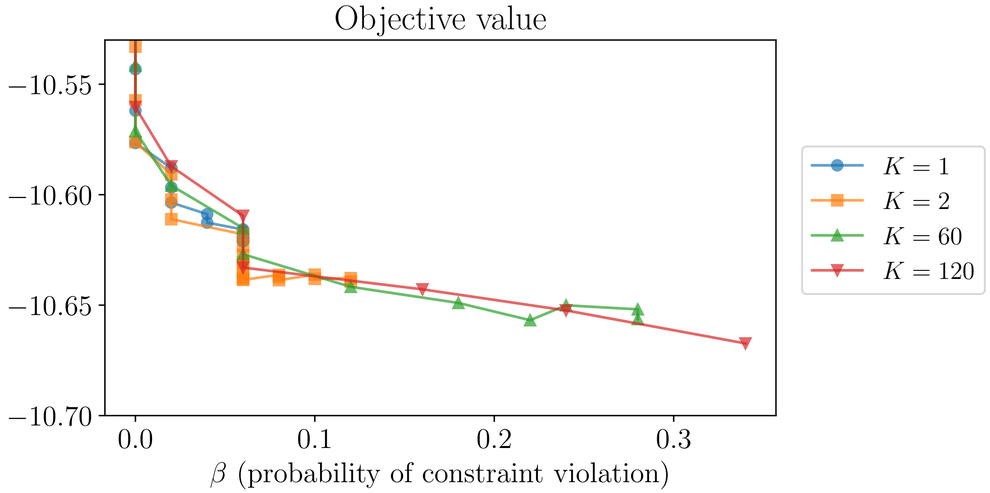
u_j ← discount rate

Capital budgeting results

$n = 20, N = 120, T = 5$



100x speedups!



2 clusters give near-optimal performance

Ongoing and future research

Mean Robust Optimization
for
Stochastic Model Predictive Control

$$
\begin{aligned}
\text{minimize} \quad & V(x_N) + \textstyle\sum_{t=0}^{N-1} x_t^T Q x_t + u_t^T R u_t \\
\text{subject to} \quad & x_{t+1} = Ax_t + Bu_t + w_t, \quad t = 0, \ldots, N-1 \\
& x_0 = x_{\text{init}} \\
& u_t \in U, \quad t = 0, \ldots, N-1 \\
& \mathbf{P}(x_t \in X) \geq 1 - \epsilon, \quad t = 0, \ldots, N
\end{aligned}
$$

Can we ensure closed-loop constraint satisfaction?

Mean Robust Optimization

- Bridge between RO and DRO

- Clustering effect
  - g affine in u → zero clustering effect!
  - g concave in u → performance bound
- Multiple orders of magnitude speedups

https://github.com/stellatogrp/mro_experiments

https://arxiv.org/abs/2207.10820

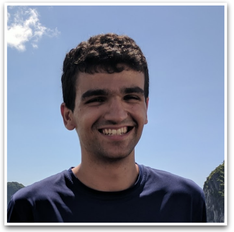
INFORMS Computing Society Student Paper Award Winner

Conclusions

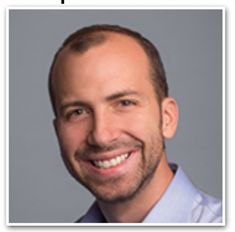
Acknowledgements

CENTER FOR STATISTICS AND MACHINE LEARNING

OSQP new features

Ian McInerney 
Vineet Bansal 
Paul Goulart 
Goran Banjac 
Rajiv Sambharya 
Mac Schaller 
Stephen Boyd 
Steven Diamond 
Akshay Agrawal 

RLQP

Jeff Ichnowski 
Paras Jain 
Michael Luo 
Ken Goldberg 
Francesco Borrelli 
Joseph Gonzales 
Ion Stoica 

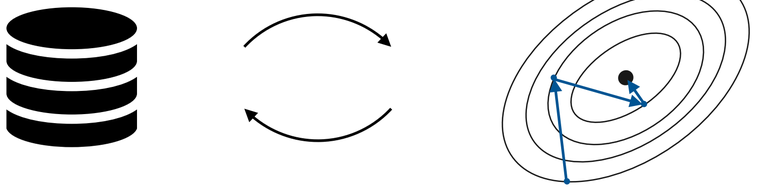
Mean Robust Optimization

Irina Wang 
Cole Becker 
Bart Van Parys 

…and many more!

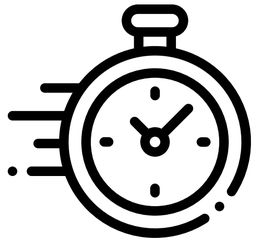
Conclusions

Data and Optimization



can help us build
more adaptive and safer autonomous systems
thanks to

Fast optimization algorithms



Robust problem formulations

Backup

Example MRO with linear constraints

$p = \infty$ $\qquad (a + Pu)^T x \leq b \longrightarrow g(u, x) = (a + Pu)^T x - b$

$$[-g]^*(z_k, x) = \sup_u z_k^T u - (a + Pu)^T x + b = \begin{cases} a^T x - b & \text{if } z_k + P^T x = 0 \\ \infty & \text{otherwise} \end{cases}$$

$$z_1 = z_2 = \cdots = z_k = -P^T x$$

minimize $\quad f(x)$

subject to $\quad \sum_{k=1}^K w_k s_k \leq 0$

$$[-g]^*(z_k, x) - z_k^T \bar{d}_k + \epsilon \|z_k\|_* \leq s_k, \quad k = 1, \ldots, K$$

minimize $\quad f(x)$

subject to $\quad a^T x - b + (P^T x)^T \bar{d} + \epsilon \|P^T x\|_* \leq 0$