# Generalization Bound and Learning Methods for Data-Driven Projections in Linear Programming

**Shinsaku Sakaue[1] and Taihei Oki[2]**

[1]The University of Tokyo, [2]Hokkaido University

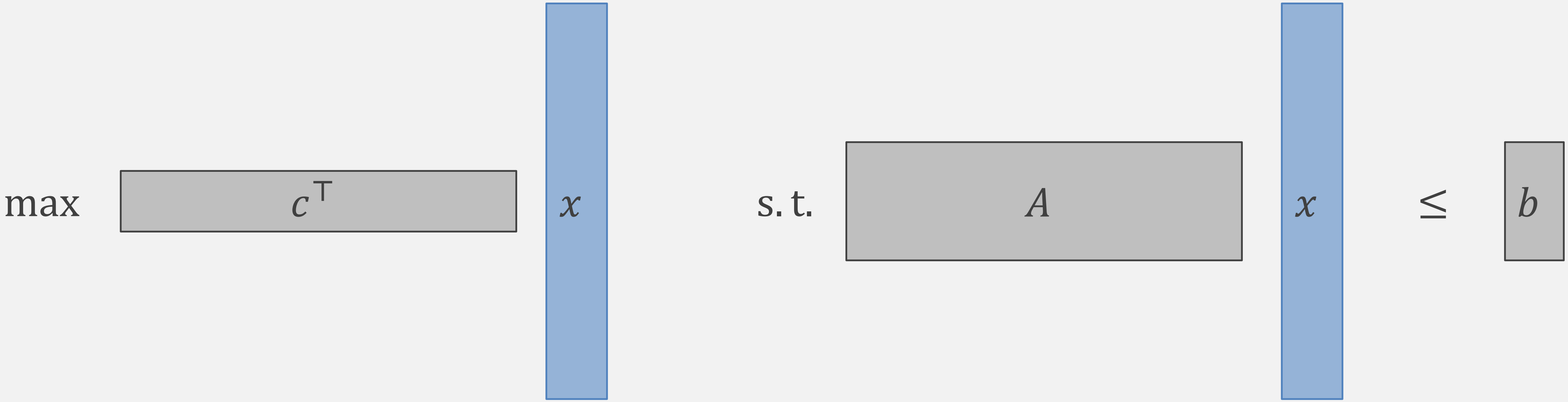
NeurIPS2024 @ Vancouver, Canada

# Linear Program

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \;\; c^\top x \qquad \text{subject to} \;\; Ax \leq b$$

We want to solve high-dimensional LPs quickly.

E.g., in transportation planning, we solve LPs with $n = \text{num.of edges in a network.}$

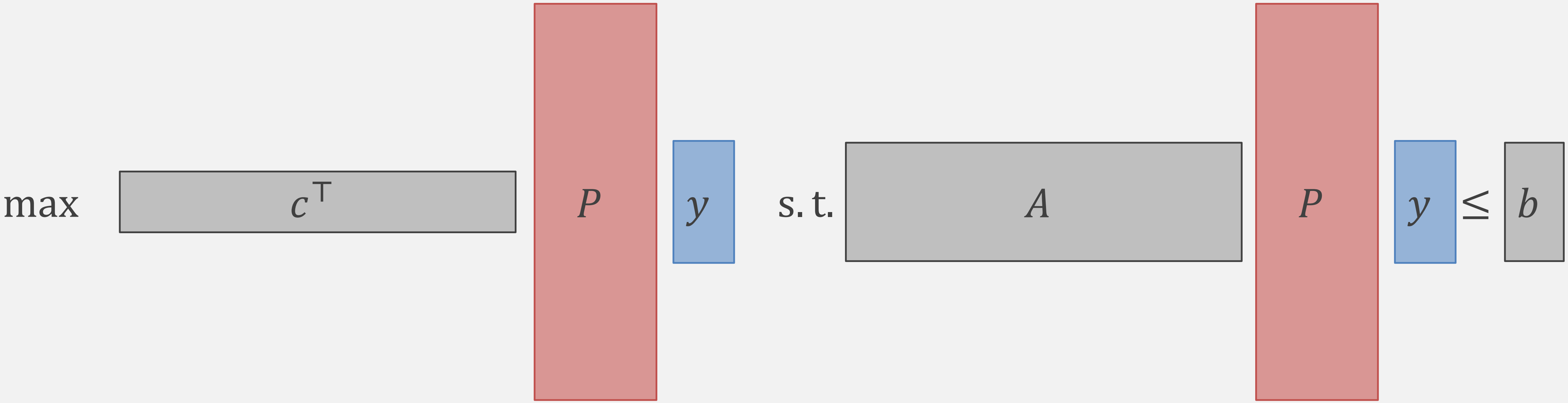


# Projection Method

$$\underset{y \in \mathbb{R}^k}{\text{maximize}} \ c^\top P y \quad \text{subject to} \ APy \leq b$$

Projection matrix $P \in \mathbb{R}^{n \times k}$ with $k \ll n$ reduces the LP dim. from $n$ to $k$.

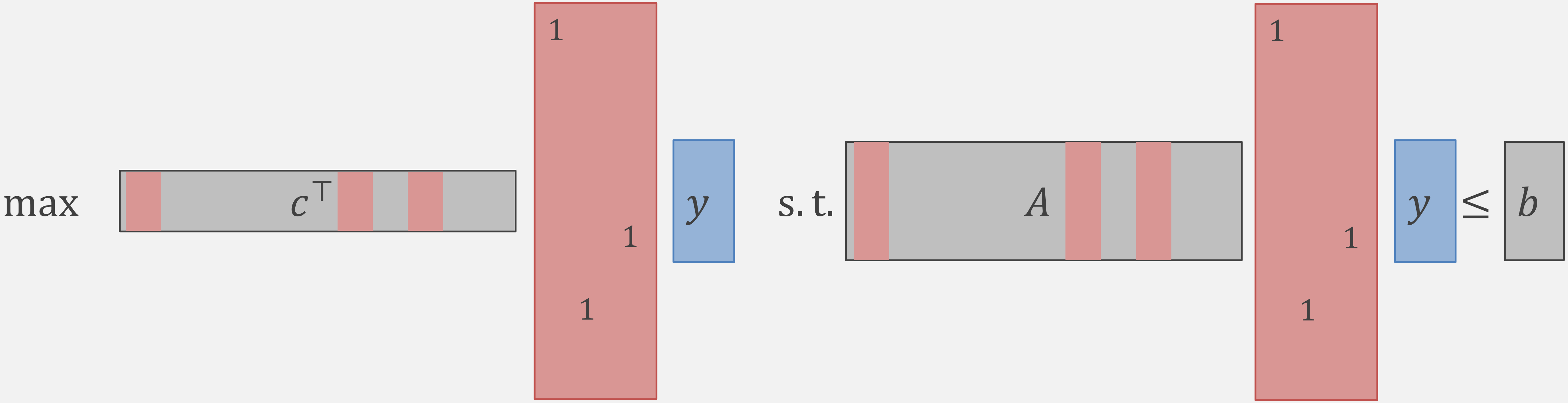
If $\text{Im}\, P$ contains good solutions, we can quickly find them by solving $k$-dim. LPs!



# Background: Random Projection

*Random projection* for LPs has been emerging (Vu et al. 2018; Poirion et al. 2023), inspired by *random sketching* in numerical linear algebra.

Akchen and Mišić (2024) used sparse $P$ for reducing LP dim. (column randomization)



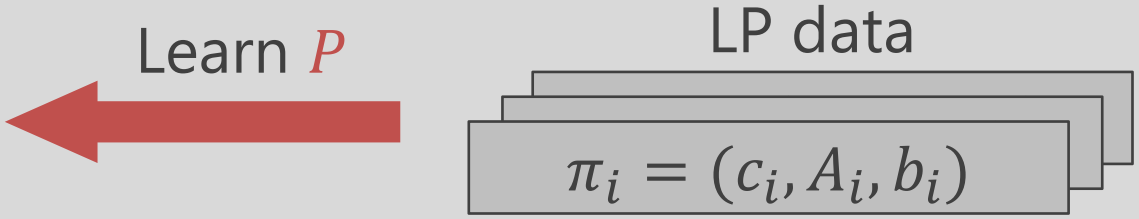

However, empirical solution quality has room for improvement (cf. Liberti et al. 2023).

# Our Approach: Data-Driven Projection

Assume data of $N$ past LP instances are available: $\pi_i = (c_i, A_i, b_i)$ for $i = 1, \dots, N$.

Learn $P$ from $\{\pi_i\}_{i=1}^{N}$ and use it when solving LPs in the future.

$$\underset{y \in \mathbb{R}^k}{\text{maximize}} \ c^\top P y \quad \text{subject to} \ A P y \le b$$




Inspired by *data-driven sketching* in numerical linear algebra (Indyk et al. 2019).

**Questions:**

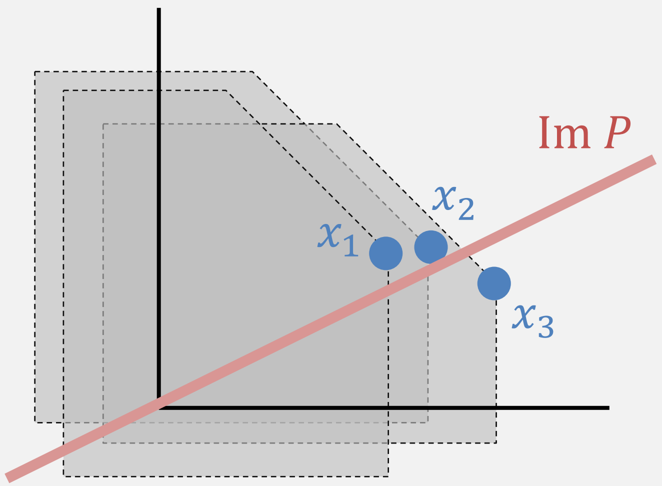
1. How to learn good $P$ in practice?
2. How much data is enough for learned $P$ to generalize to future LPs?

# Learning Method 1: PCA

Solve training LPs $\pi_i = (c_i, A_i, b_i)$ to find opt. sol. $x_i \in \operatorname{argmax}\{c^\top x \mid Ax \leq b\}$.

$\mathrm{Im}\, P$ should cover a $k$-dim subspaces close to $x_i$'s.

Apply PCA to $(x_1, \dots, x_N)^\top$ so that $x_i \approx P y_i$ holds for some $y_i \in \mathbb{R}^k$.



# Learning Method 2: Gradient Ascent

Consider improving $u(P, \pi_i)$ directly by gradient-based updates.

Under some regularity conditions, we can compute the gradient w.r.t. $P$:

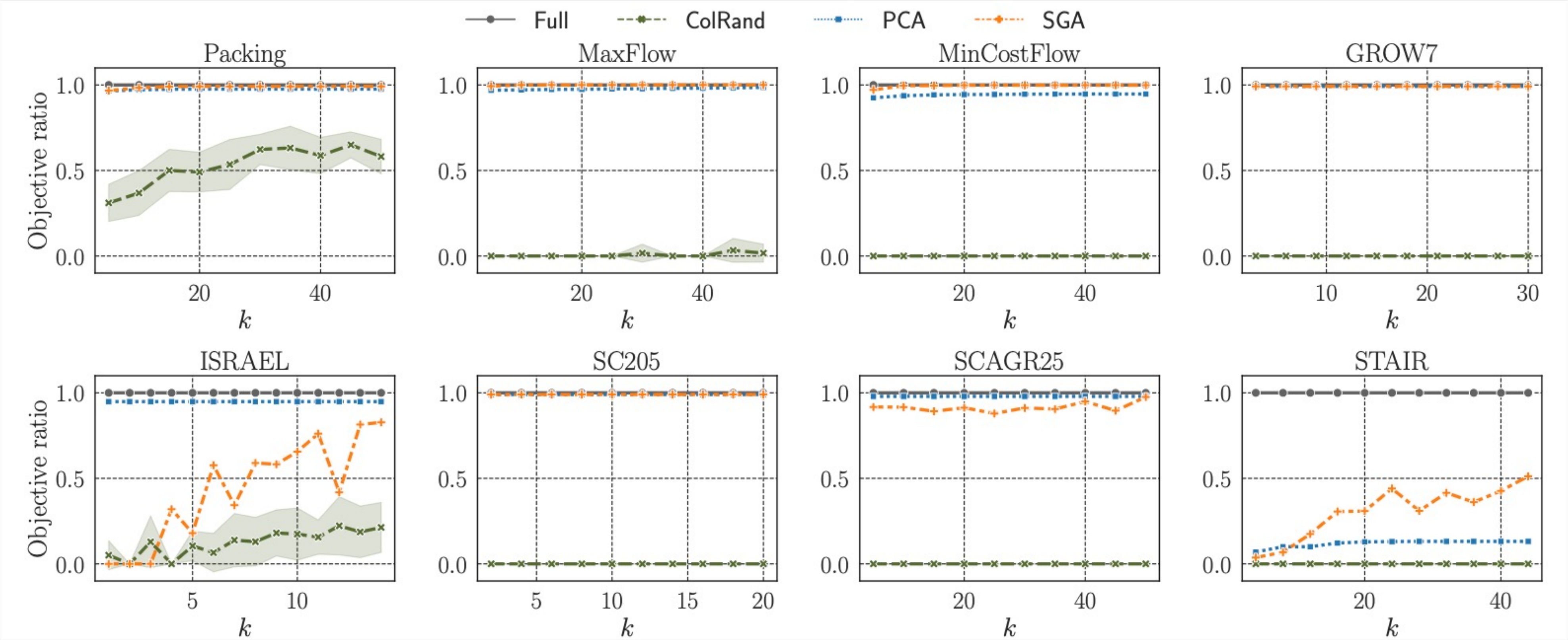
$$\nabla u(P, \pi_i) = \nabla \max\{c_i^\top P y \mid A_i P y \leq b_i\}$$

via the implicit function theorem.

Apply stochastic gradient ascent to maximize $\frac{1}{N}\sum_{i=1}^{N} u(P, \pi_i)$.

# Experiments

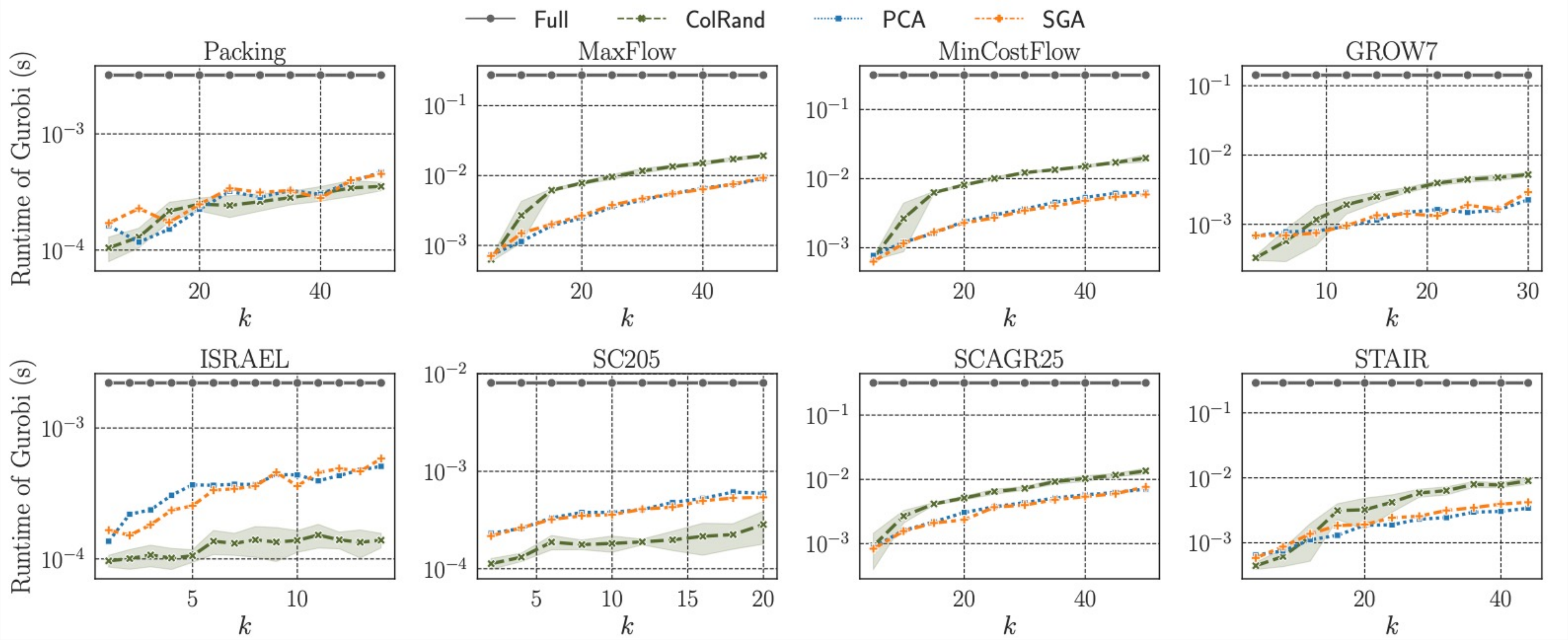
Full = w/o projection; ColRand = random projection of Akchen and Mišić (2024).
PCA and SGA learn $P$ from data. All LPs are solved with Gurobi.




PCA and SGA lead to near optimal objectives in most datasets, outperforming ColRand.

# Experiments

Full = w/o projection; ColRand = random projection of Akchen and Mišić (2024).
PCA and SGA learn $P$ from data. All LPs are solved with Gurobi.




Projection-based methods are much faster than Full. PCA and SGA enable fast "and" accurate solving.

# Generalization Bound

Assume LP instances $\pi = (c, A, b) \in \Pi$ are drawn from a distribution $\mathcal{D}$.

Define $u(P, \pi) \coloneqq \max\{c^\top P y \mid APy \leq b\}$ and $\mathcal{U} \coloneqq \{u(P, \cdot) : \Pi \to \mathbb{R} \mid P \in \mathbb{R}^{n \times k}\}$.

**Uniform convergence** (Pollard 1984): given $\{\pi_i\}_{i=1}^N \sim \mathcal{D}^N$

*for all* $P \in \mathbb{R}^{n \times k}$, w.h.p., $\left| \frac{1}{N} \sum_{i=1}^N u(P, \pi_i) - \mathbb{E}_{\pi \sim \mathcal{D}}[u(P, \pi)] \right| \lesssim \sqrt{\frac{\mathrm{pdim}(\mathcal{U})}{N}}$

Empirical objective values attained with $P$ **at hand.**

Expected objective values attained with $P$ **in the future.**

Pseudo-dimension (complexity) of $\mathcal{U}$.

The bound holds *uniformly for all* $P \in \mathbb{R}^{n \times k}$, regardless of how it is learned!

Common idea in *data-driven algorithm design* (Gupta–Roughgarden 2017; Balcan 2021).

# Pseudo-Dimension Bounds

**Theorem** $\mathrm{pdim}(\mathcal{U}) = \tilde{O}(nk^2)$ (and $\Omega(nk)$).

Proof idea (inspired by Balcan et al. 2022)

$$\mathrm{pdim}(\mathcal{U}) = \max N \ \text{s.t.}\ \exists \pi_1 \dots, \pi_N \in \Pi,\ \exists t_1 \dots, t_N \in \mathbb{R},\ \left|\left\{\left(\mathbb{1}_{u(P,\pi_i)>t_i}\right)_{i=1}^N \,\middle|\, P \in \mathbb{R}^{n\times k}\right\}\right| = 2^N.$$

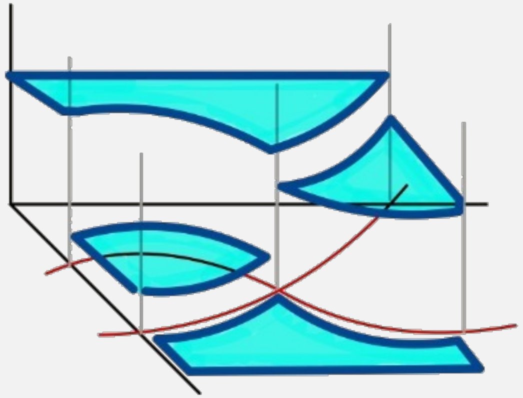
$u(P, \pi_i)$'s are attained at vertices; num. of vertices $\lesssim (\#\text{constraints})^k$.

"$u(P, \pi_i) > t_i$?" is determined by inequalities of "obj. at some vertex $> t_i$?" which are polynomials of $P \in \mathbb{R}^{n\times k}$ of degree $O(k)$ due to Cramer's rule.

By Warren's theorem,

$$\left|\left\{\left(\mathbb{I}(u(P,\pi_i) > t_i)\right)_{i=1}^N \,\middle|\, P \in \mathbb{R}^{n\times k}\right\}\right| \lesssim \left(N(\#\text{constraints})^k k/(nk)\right)^{nk}.$$

Solving $\left(N(\#\text{constraints})^k/(nk)\right)^{nk} \leq 2^N$ implies the $\tilde{O}(nk^2)$ bound.



Polynomials of $P \in \mathbb{R}^{n\times k}$ partition $\mathbb{R}^{n\times k}$ into cells. In each cell, outcomes of "$u(P, \pi_i) > t_i$?" remain the same.