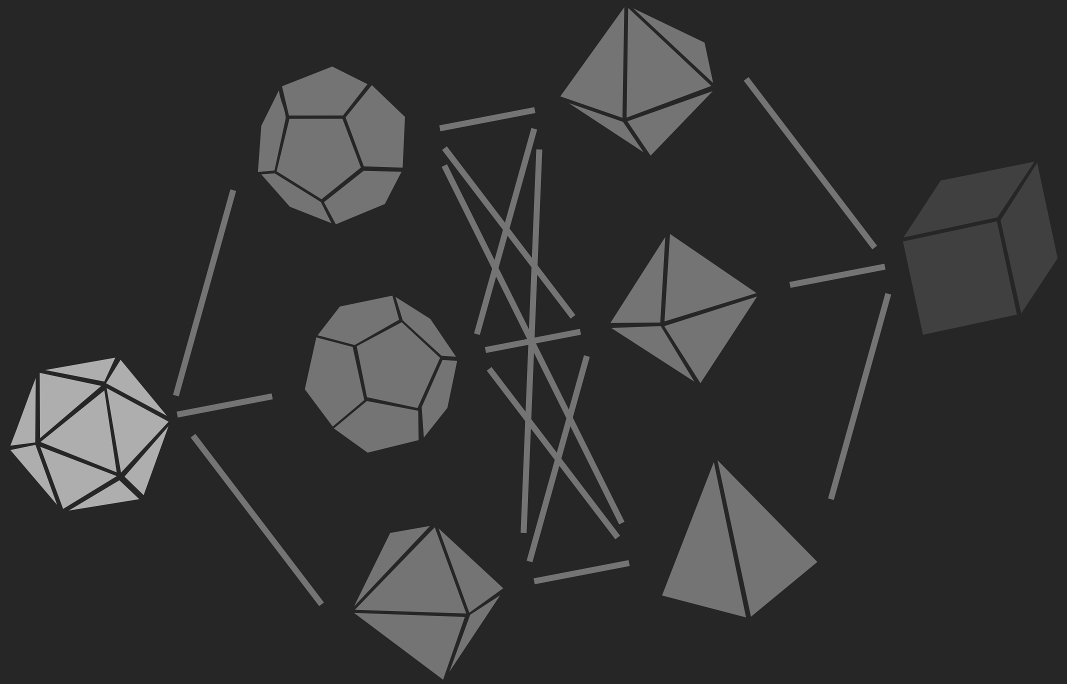
CPAIOR 2024 | Session 12: Decision-focused Learning

# CaVE

## A Cone-Aligned Approach for Fast Predict-then-Optimize



Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

Presented by Bo Tang

Uppsala, May 30, 2024

# Authors

**Bo Tang**

PhD Candidate
Department of Mechanical & Industrial Engineering, University of Toronto

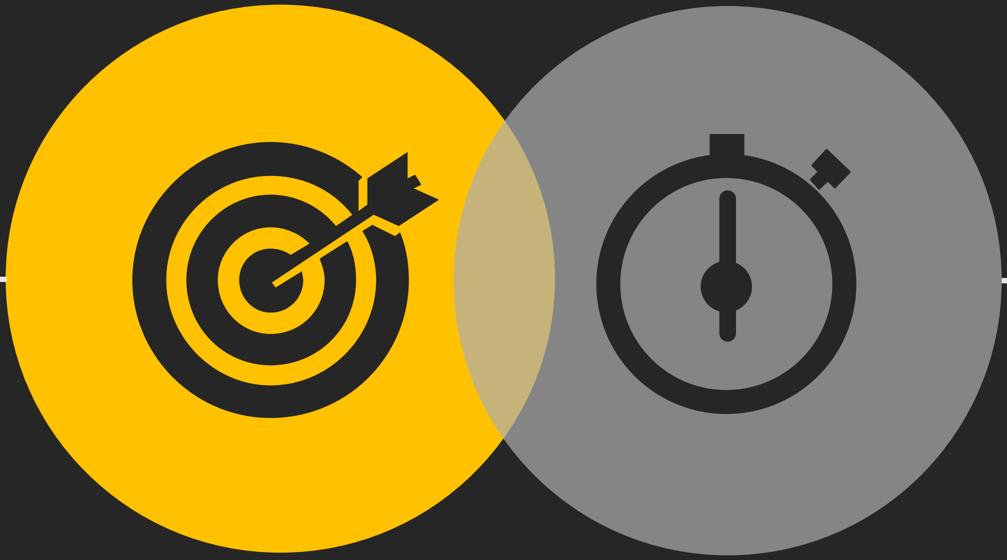
**Elias B. Khalil**

Assistant Professor
Department of Mechanical & Industrial Engineering, University of Toronto

SCALE AI Research Chair
Data-Driven Algorithms for Modern Supply Chains

# Introduction

CaVE (Cone-aligned Vector Estimation) is a **Decision-focused Learning / End-to-end Predict-then-optimize** approach for **Binary Linear Programs** (BLPs).



**End-to-End:**

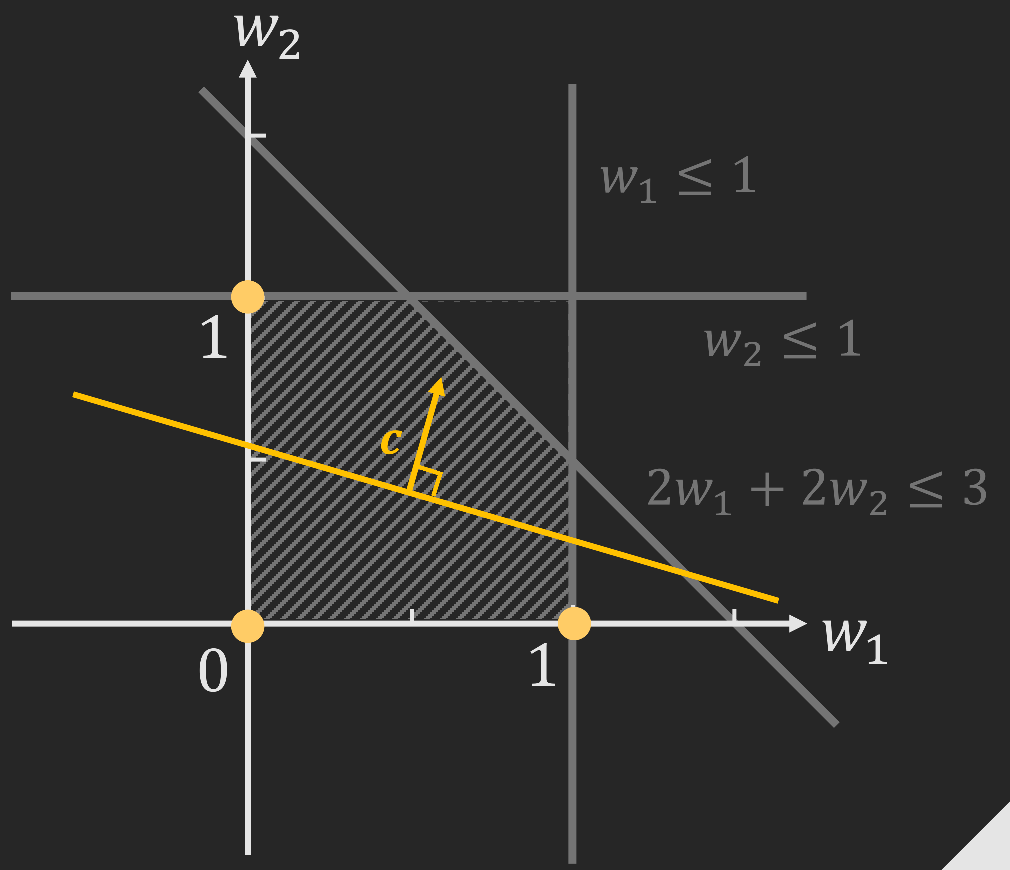
The loss function of CaVE focuses on decision quality.

**Efficiency:**

The algorithm of CaVE utilizes non-negative least squares (NNLSs) instead of solving BLPs.

# Notation

$$\min_{\underbrace{\boldsymbol{w}}_{\text{Decision Variables}}} \left\{ \overbrace{\boldsymbol{c}^\top \boldsymbol{w}}^{\text{Linear Objective Function}} : \underbrace{\boldsymbol{w} \in \boldsymbol{S}}_{\text{Feasible Region}}, \overbrace{\boldsymbol{w} \in \{0,1\}}^{\text{Binary Domain}} \right\}$$



# Predict, then Optimize

$$\min_{\boldsymbol{w}}\{\, \boldsymbol{c}_1{}^{\top}\boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \,\}$$

$$\min_{\boldsymbol{w}}\{\, \boldsymbol{c}_2{}^{\top}\boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \,\}$$

$$\min_{\boldsymbol{w}}\{\, \boldsymbol{c}_3{}^{\top}\boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \,\}$$

$$\ddots$$

# Predict, then Optimize

Unknown Coefficients

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_1^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_2^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_3^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$\ddots$

Identical Constraints

# Predict, then Optimize

Unknown Coefficients

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_1^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_2^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_3^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$\ddots$

Identical Constraints

Observed Feature Vectors

$\boldsymbol{x}_1$

$\boldsymbol{x}_2$

$\boldsymbol{x}_3$

$\vdots$

# Predict, then Optimize

Unknown Coefficients

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_1^\top \boldsymbol{w} : \boldsymbol{w} \in S\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_2^\top \boldsymbol{w} : \boldsymbol{w} \in S\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_3^\top \boldsymbol{w} : \boldsymbol{w} \in S\}$$

$\ddots$

Identical Constraints

Observed Feature Vectors

$\boldsymbol{x}_1$

$\boldsymbol{x}_2$

$\boldsymbol{x}_3$

$\vdots$

$\hat{\boldsymbol{c}} = g(\boldsymbol{x}; \boldsymbol{\theta})$

**Prediction**

# Predict, then Optimize

Unknown Coefficients

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_1^\top \boldsymbol{w} : \boldsymbol{w} \in S \}$$

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_2^\top \boldsymbol{w} : \boldsymbol{w} \in S \}$$

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_3^\top \boldsymbol{w} : \boldsymbol{w} \in S \}$$

$\ddots$

Identical Constraints

Predicted Coefficients

Model Parameters

$$\hat{\boldsymbol{c}} = g(\boldsymbol{x}; \boldsymbol{\theta})$$

**Prediction**

Observed Feature Vectors

$\boldsymbol{x}_1$

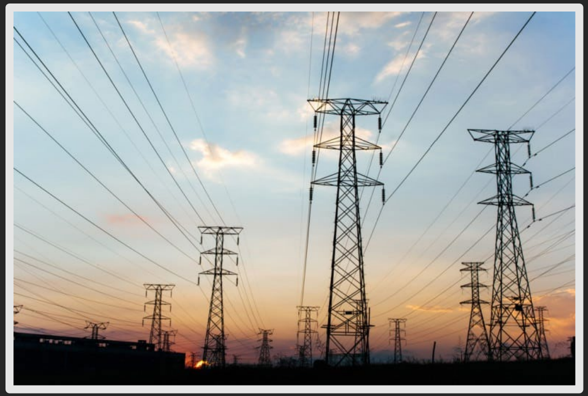
$\boldsymbol{x}_2$

$\boldsymbol{x}_3$

$\vdots$

# Examples


❖Vehicle Routing



❖Energy Scheduling

❖Portfolio Optimization

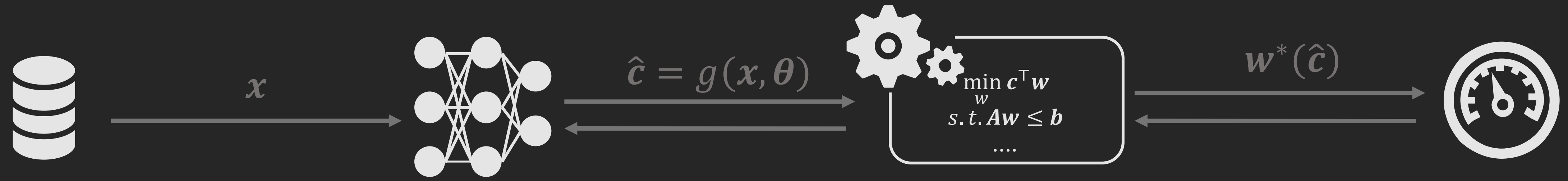
# Examples


- Vehicle Routing
- Energy Scheduling
- Portfolio Optimization

**Unknown Costs:** Travel Time, Electricity Prices, Asset Returns

**Observed Features:** Distance, Time, Weather, Financial Factors...

# Decision-focused Learning



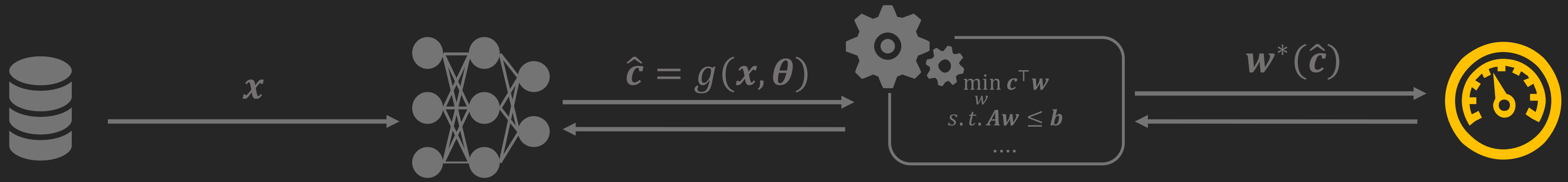

Dataset $\mathcal{D}$ with data points $(\boldsymbol{x}, \boldsymbol{c})$

Prediction model $g(\boldsymbol{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$

Optimization solver $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = \operatorname{argmin}_{\boldsymbol{w} \in S} \hat{\boldsymbol{c}}^\top \boldsymbol{w}$

Loss function $\mathcal{L}(\cdot)$ to measure decision error

# Decision-focused Learning




Dataset $\mathcal{D}$ with data points $(x, c)$

Prediction model $g(x, \theta)$ with parameters $\theta$

Optimization solver $w^*(\hat{c}) = \underset{w \in S}{\operatorname{argmin}}\, \hat{c}^{\top} w$

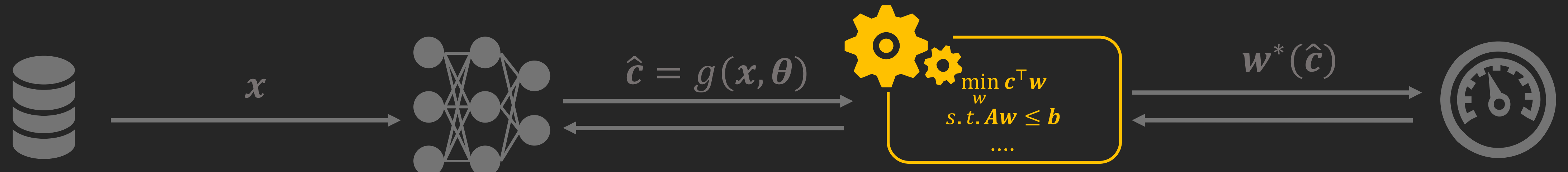
Loss function $\mathcal{L}(\cdot)$ to measure decision error

**e.g.**

$$\mathcal{L}_{\text{Regret}}(\hat{c}, c) = c^{\top} w^*(\hat{c}) - c^{\top} w^*(c)$$

$$\mathcal{L}_{\text{Square}}(\hat{c}, c) = \frac{1}{2} \| w^*(c) - w^*(\hat{c}) \|_2^2$$

# Decision-focused Learning




Dataset $\mathcal{D}$ with data points $(\boldsymbol{x}, \boldsymbol{c})$

Prediction model $g(\boldsymbol{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$

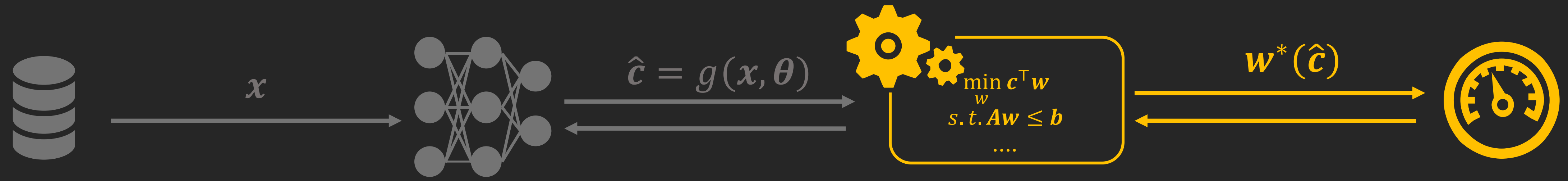
Optimization solver $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = \underset{\boldsymbol{w} \in S}{\operatorname{argmin}}\, \hat{\boldsymbol{c}}^{\top} \boldsymbol{w}$

Loss function $\mathcal{L}(\cdot)$ to measure decision error

**Computational Bottleneck:**
All state-of-the-art methods require repeated solving during the iteration.

**1,000 Dataset Instances**
$\times$
**20 Training Epochs**
$\Downarrow$
**≥20,000 Solving!!!**

# Decision-focused Learning




Dataset $\mathcal{D}$ with data points $(x, c)$

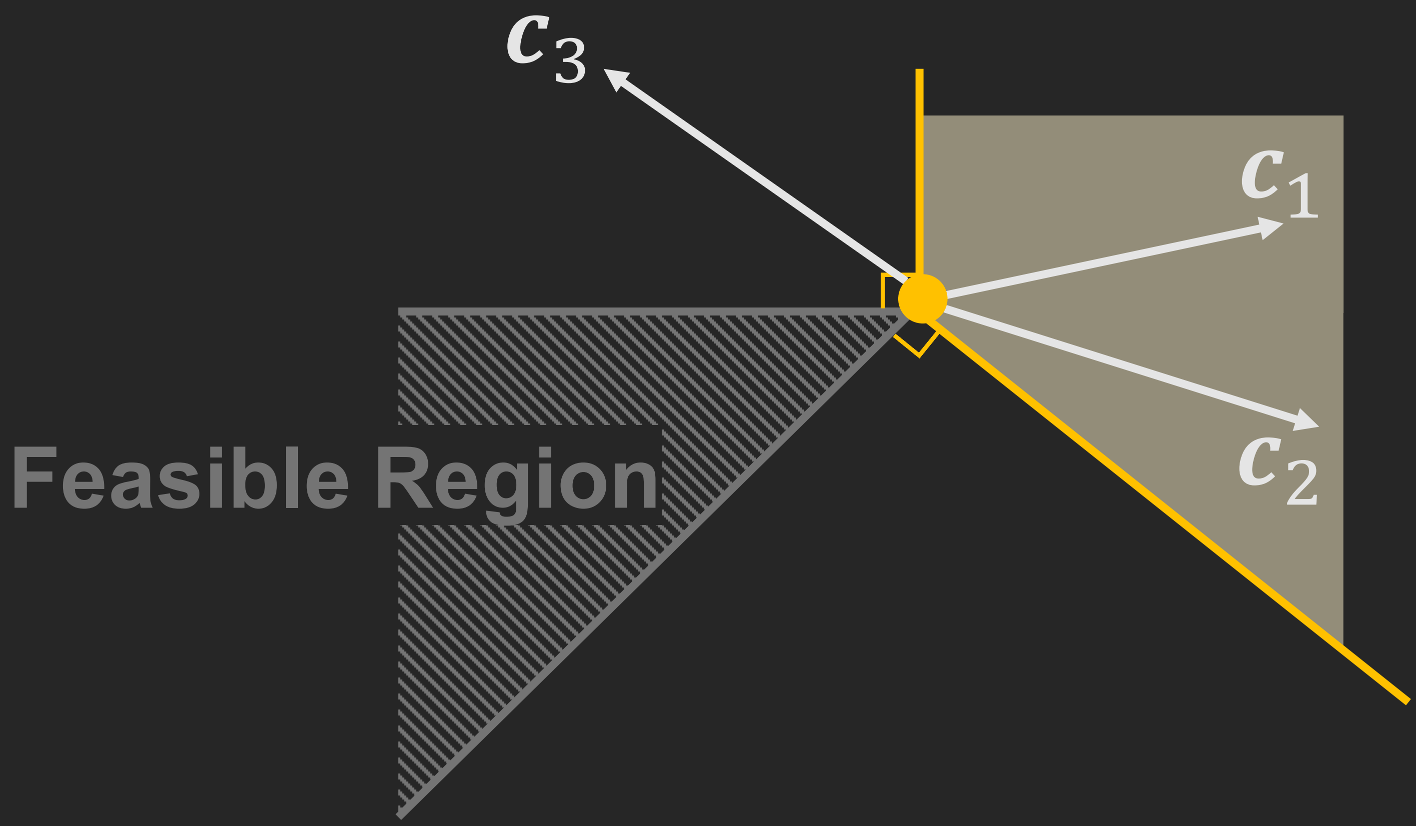
Prediction model $g(x, \theta)$ with parameters $\theta$

Optimization solver $w^*(\hat{c}) = \operatorname{argmin}_{w \in S} \hat{c}^\top w$

Loss function $\mathcal{L}(\cdot)$ to measure decision error

**CaVE:**
Replace the original optimization with projection (quadratic programming).

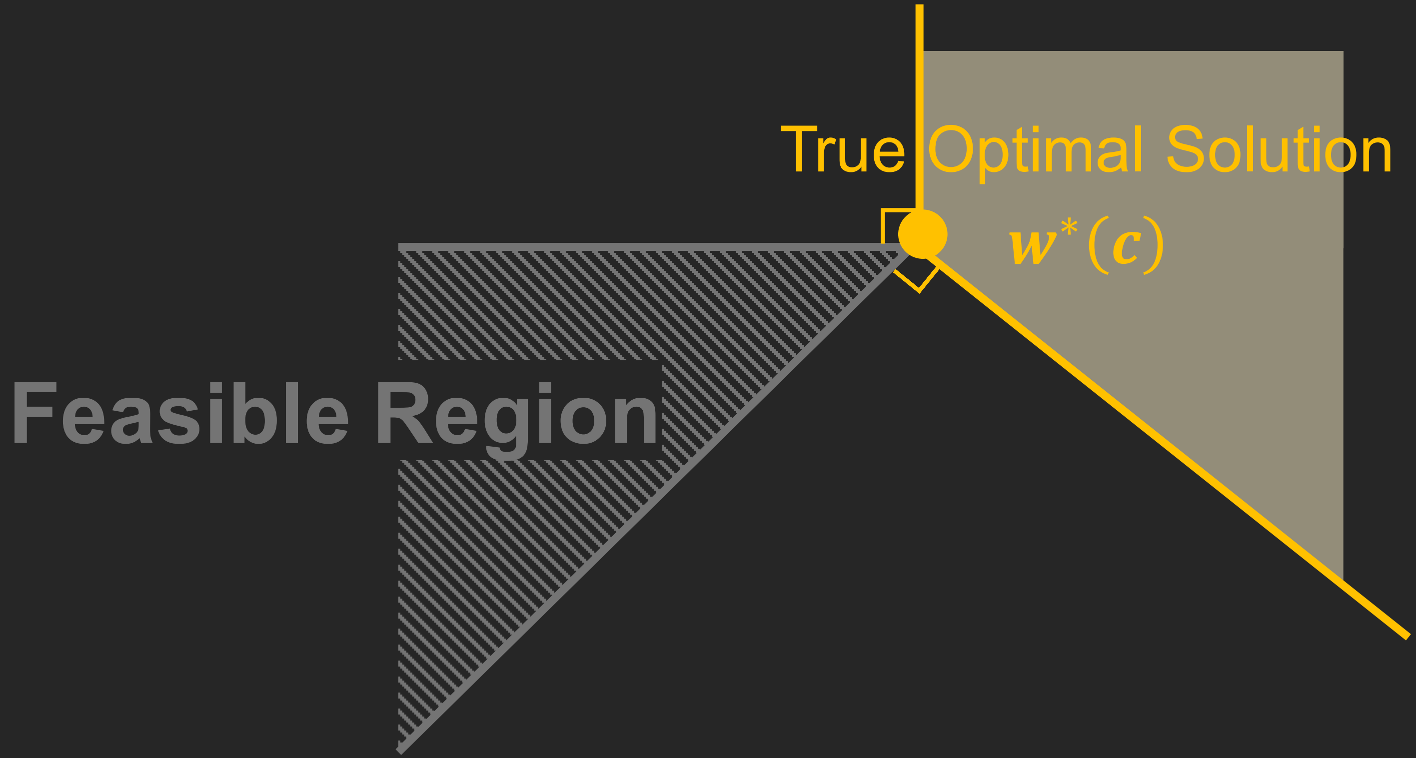
# Geometric Intuition




For LP, cost vectors in the same **normal cone** have the same optimal solutions.

$$w^*(c_1) = w^*(c_2) \neq w^*(c_3)$$

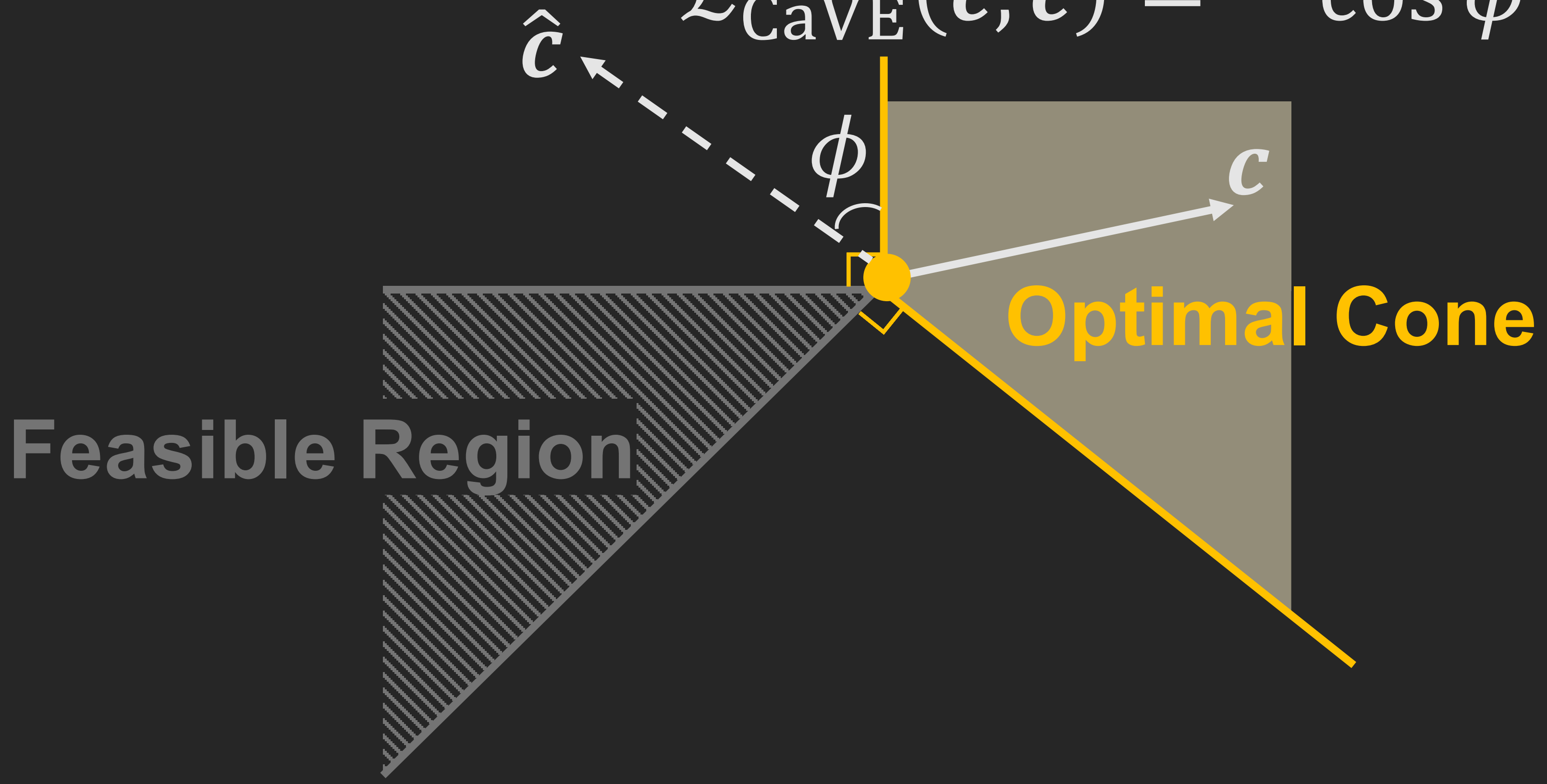
# Geometric Intuition




For LP, **optimal cone** $\mathcal{C}^*(c)$ is defined as the normal cone at the true optimal solution $w^*(c)$.

# Similarity Loss Function

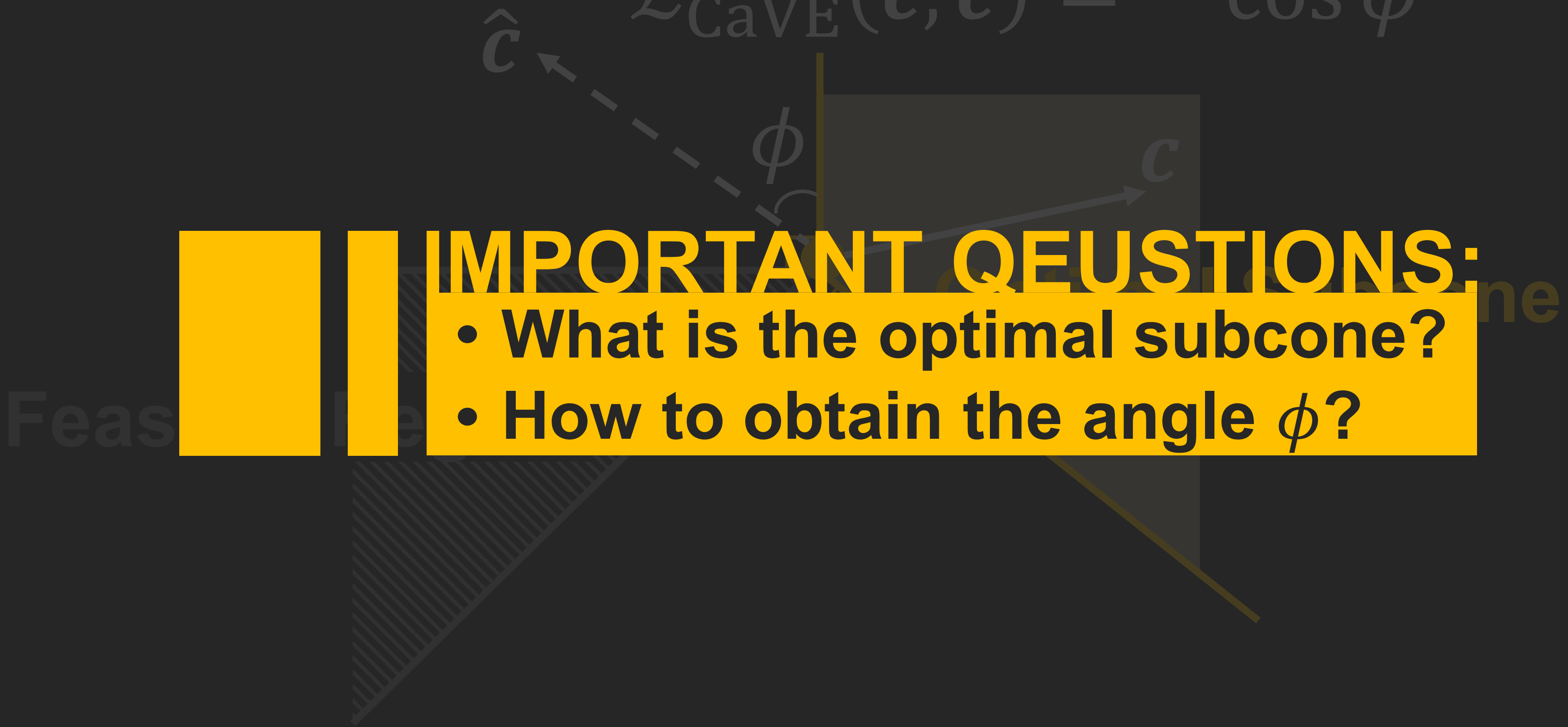
$$\mathcal{L}_{\mathrm{CaVE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = -\cos\phi$$




The loss of CaVE employs cosine similarity to minimize the **angle $\phi$** between the **predicted costs $\hat{c}$** and **optimal subcone** (a subset of **optimal cone**).

# Similarity Loss Function

$$\mathcal{L}_{\text{CaVE}}(\hat{c}, c) = -\cos\phi$$

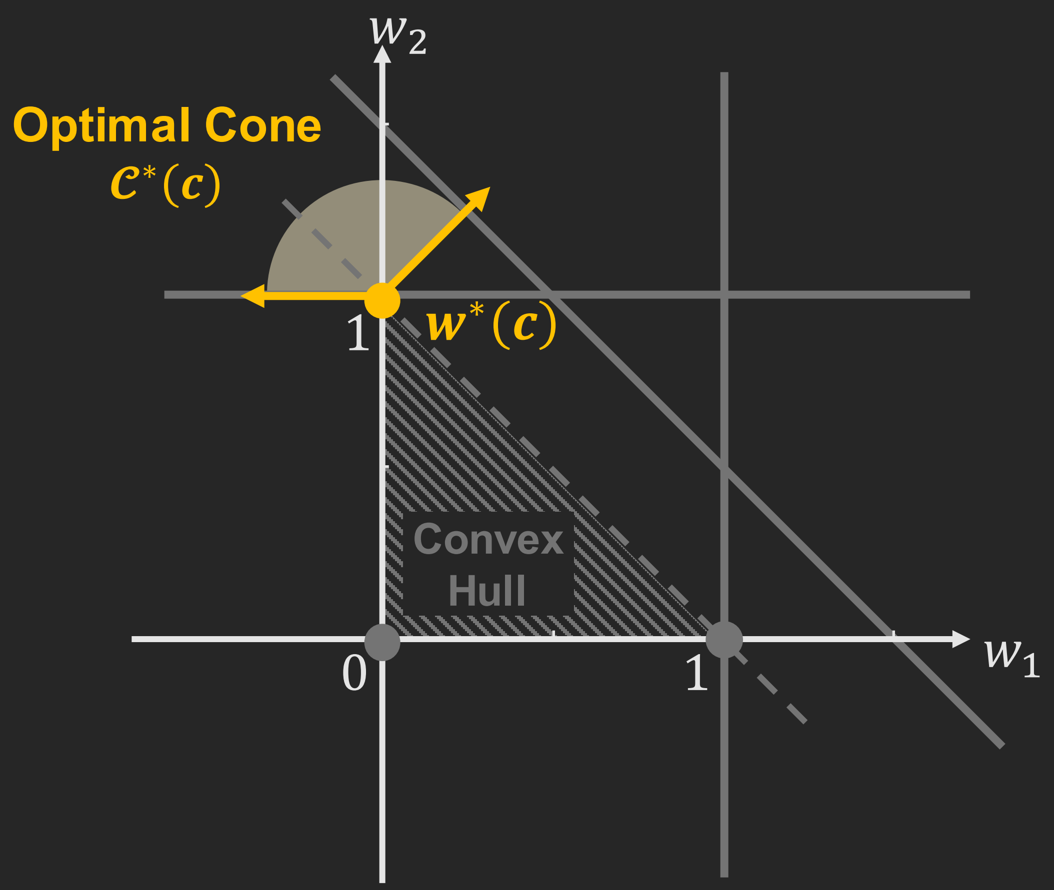



**IMPORTANT QEUSTIONS:**

- **What is the optimal subcone?**
- **How to obtain the angle $\phi$?**

The loss of CaVE employs cosine similarity to minimize the **angle $\phi$** between the **predicted costs $\hat{c}$** and **optimal subcone** (a subset of **optimal cone**).

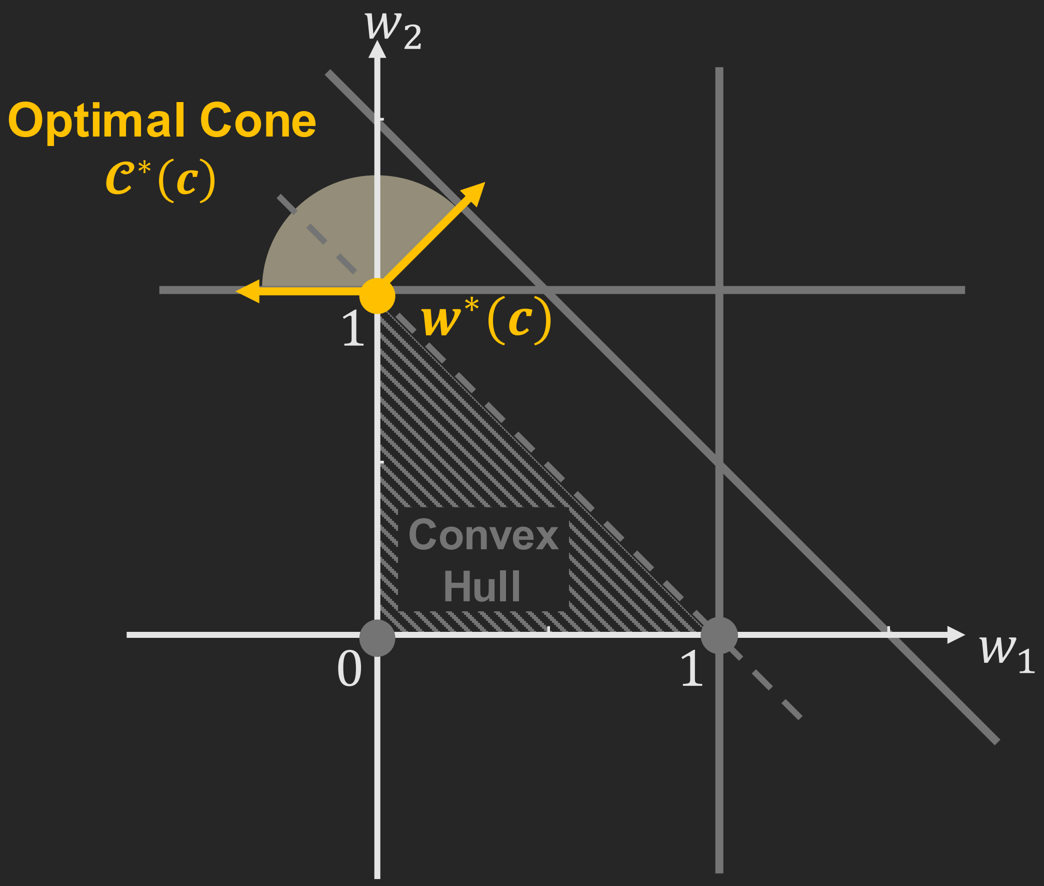
# Optimal Subcone




Binary Linear Program

1. For ILP, the normal cone to the convex hull at the optimal solution $w^*(c)$ is defined as optimal cone $\mathcal{C}^*(c)$.

# Optimal Subcone




Binary Linear Program

1. For ILP, the normal cone to the convex hull at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal cone $\mathcal{C}^*(\boldsymbol{c})$.

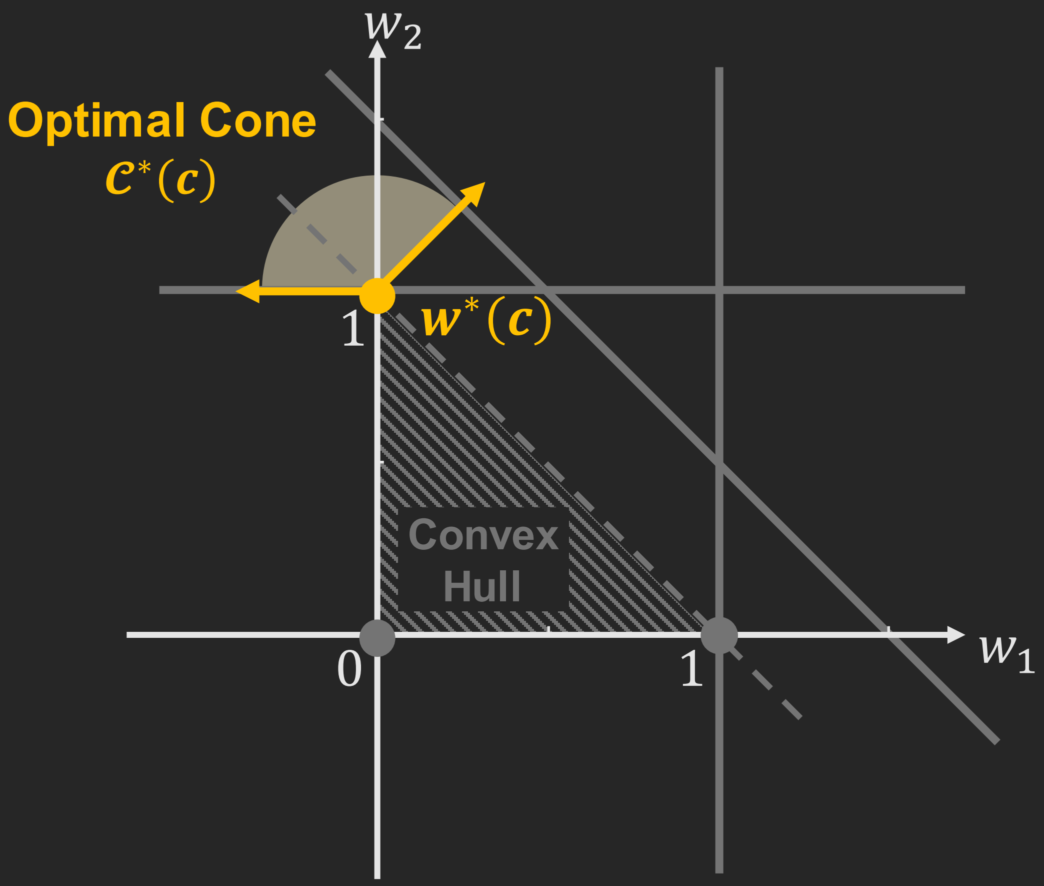
2. $\forall \boldsymbol{c}' \in \mathcal{C}^*(\boldsymbol{c}), \boldsymbol{w}^*(\boldsymbol{c}') = \boldsymbol{w}^*(\boldsymbol{c})$. Cost vectors yield the same optimal solution if and only if they are in the same optimal cone.

# Optimal Subcone

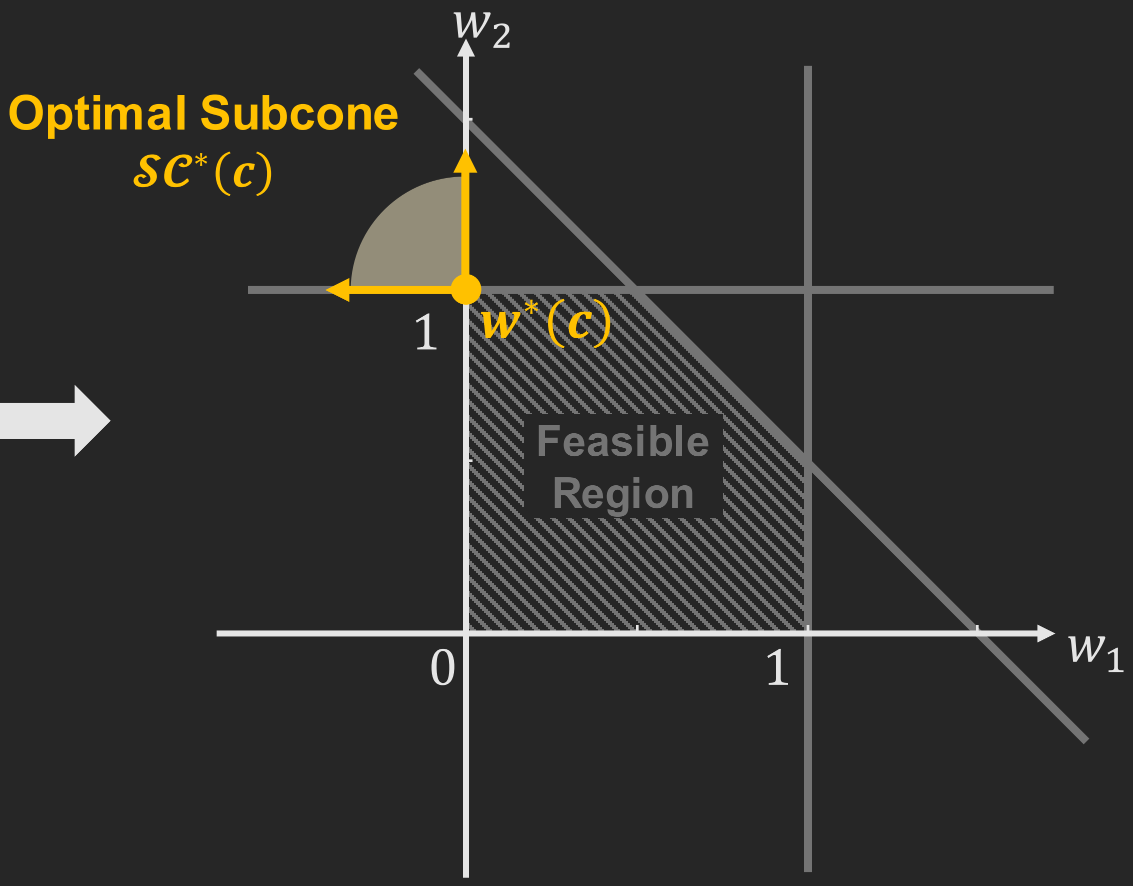

Binary Linear Program

1. For ILP, the normal cone to the convex hull at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal cone $\boldsymbol{\mathcal{C}}^*(\boldsymbol{c})$.
2. $\forall \boldsymbol{c}' \in \boldsymbol{\mathcal{C}}^*(\boldsymbol{c}), \boldsymbol{w}^*(\boldsymbol{c}') = \boldsymbol{w}^*(\boldsymbol{c})$. Cost vectors yield the same optimal solution if and only if they are in the same optimal cone.
3. However, for ILP, obtaining the convex hull is NOT trivial. e.g., Cutting Plane method...

# Optimal Subcone



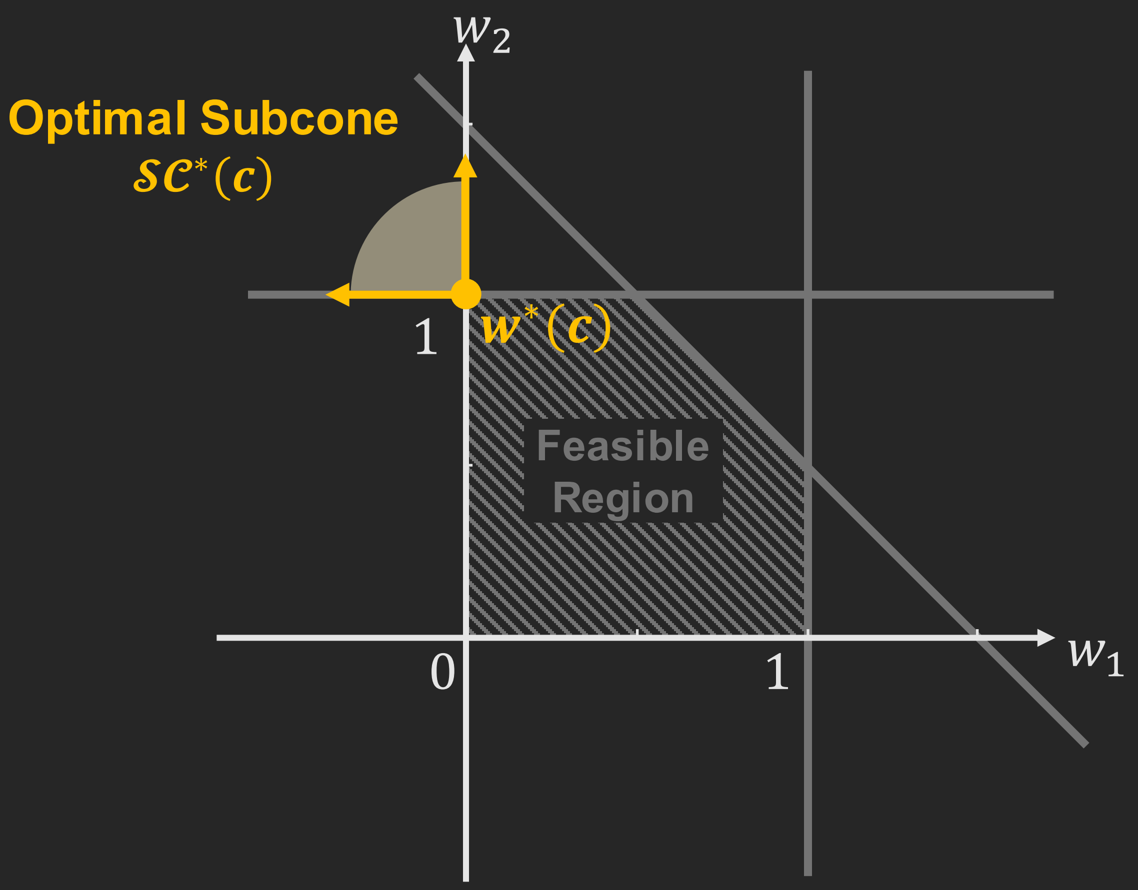

Binary Linear Program




Linear Relaxation

# Optimal Subcone

1. For BLP, the normal cone to the feasible region of linear relaxation at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal subcone $\mathcal{SC}^*(\boldsymbol{c})$.

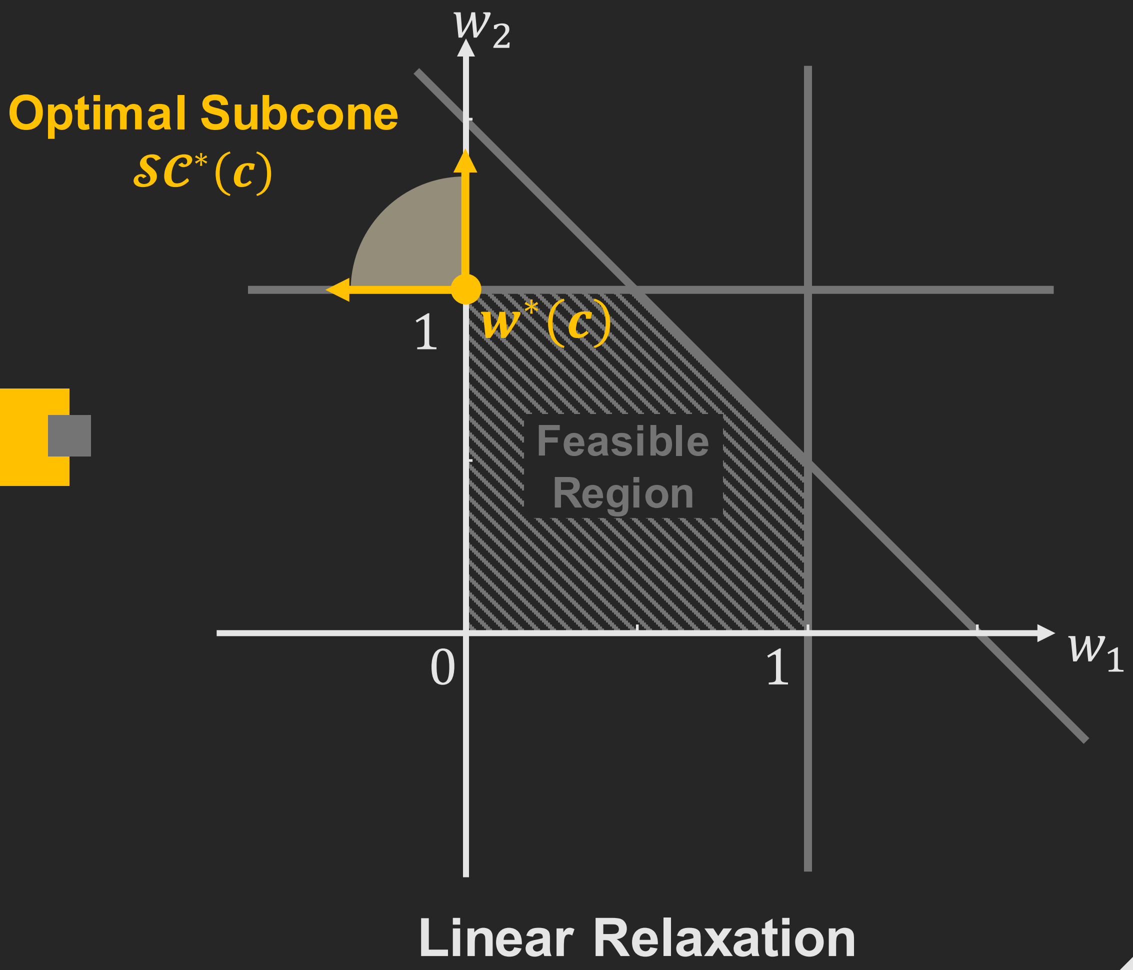



Linear Relaxation

# Optimal Subcone

1. For BLP, the normal cone to the feasible region of linear relaxation at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal subcone $\mathcal{SC}^*(\boldsymbol{c})$.

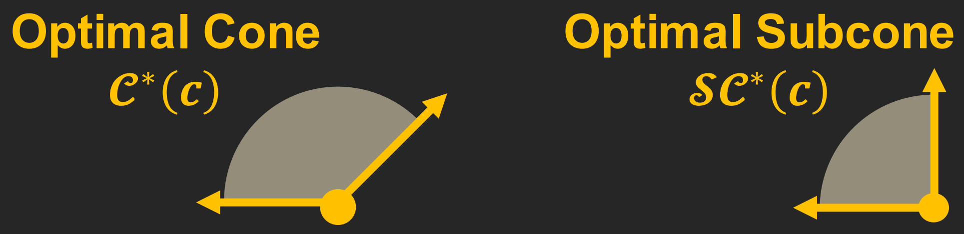
**Note: This does not apply to general ILP!**

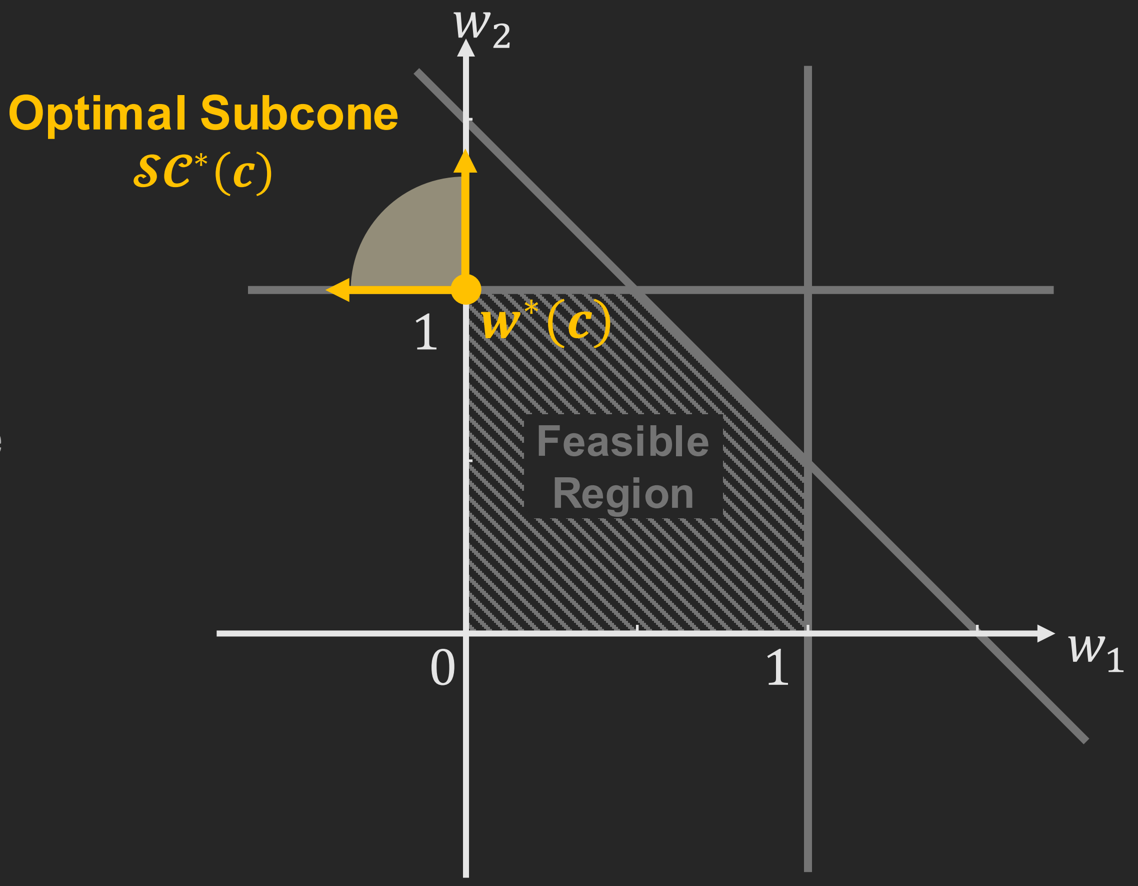


# Optimal Subcone

1. For BLP, the normal cone to the feasible region of linear relaxation at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal subcone $\mathcal{SC}^*(\boldsymbol{c})$.

2. $\mathcal{SC}^*(\boldsymbol{c}) \subseteq \mathcal{C}^*(\boldsymbol{c})$. Thus, cost vectors yield the same optimal solution if they are in the same optimal subcone.




Linear Relaxation

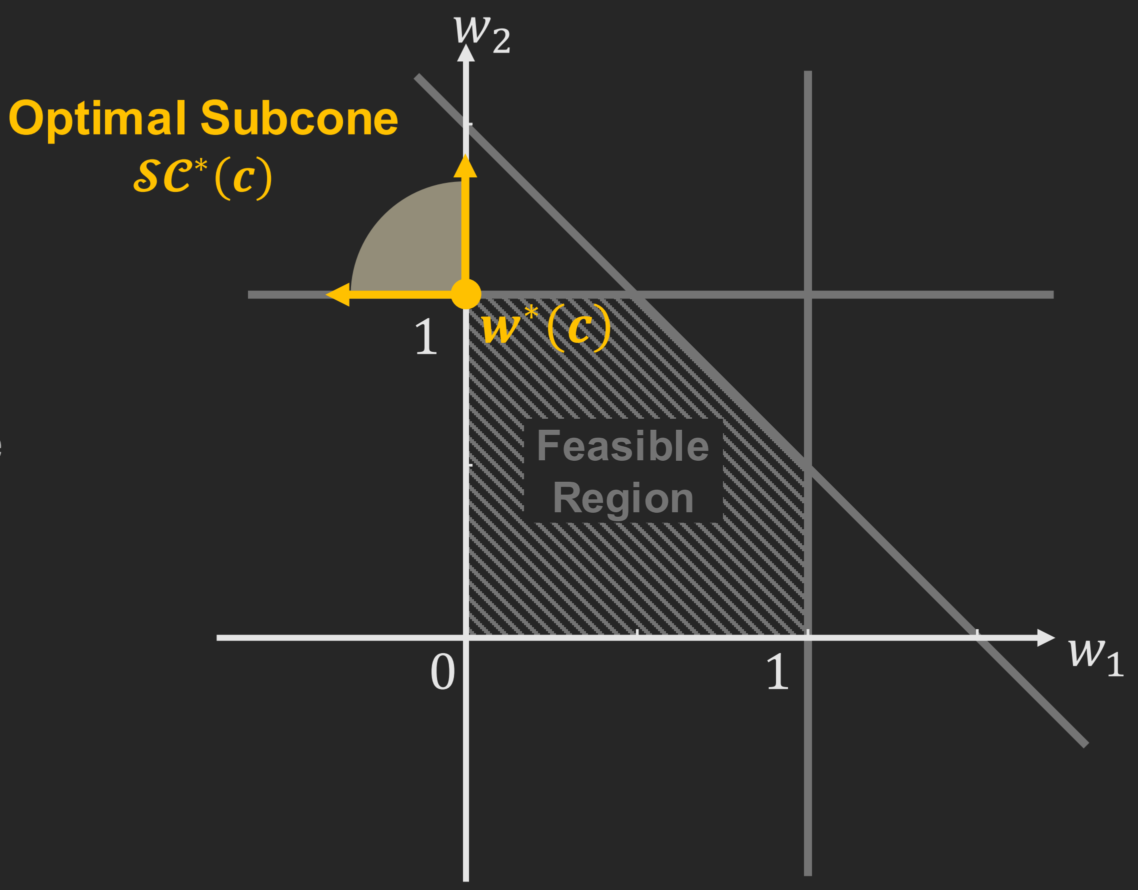
# Optimal Subcone

1. For BLP, the normal cone to the feasible region of linear relaxation at the optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$ is defined as optimal subcone $\mathcal{SC}^*(\boldsymbol{c})$.

2. $\mathcal{SC}^*(\boldsymbol{c}) \subseteq \mathcal{C}^*(\boldsymbol{c})$. Thus, cost vectors yield the same optimal solution if they are in the same optimal subcone.

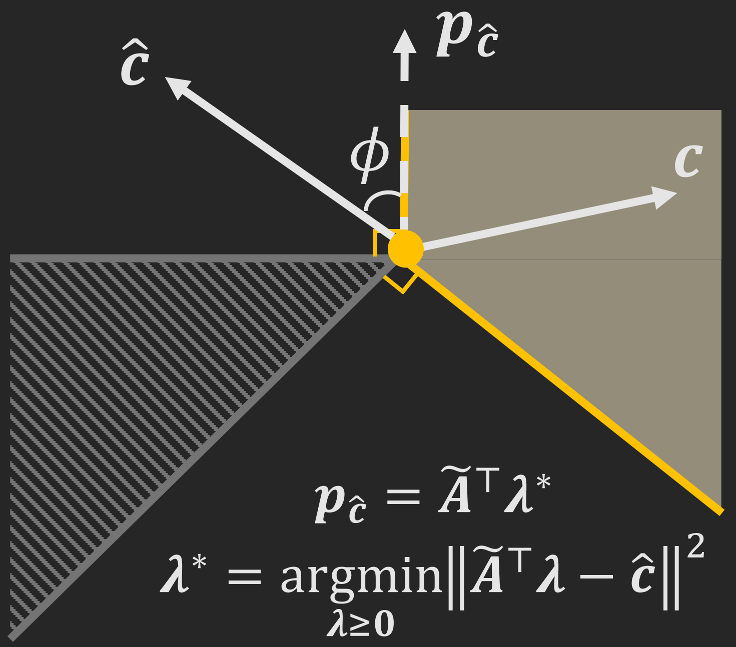
3. The optimal subcone is the conic combination of tight constraints, which is trivial.

$$\tilde{A}(\boldsymbol{c}): \tilde{A}(\boldsymbol{c})^\top \boldsymbol{w}^*(\boldsymbol{c}) = \boldsymbol{b}$$



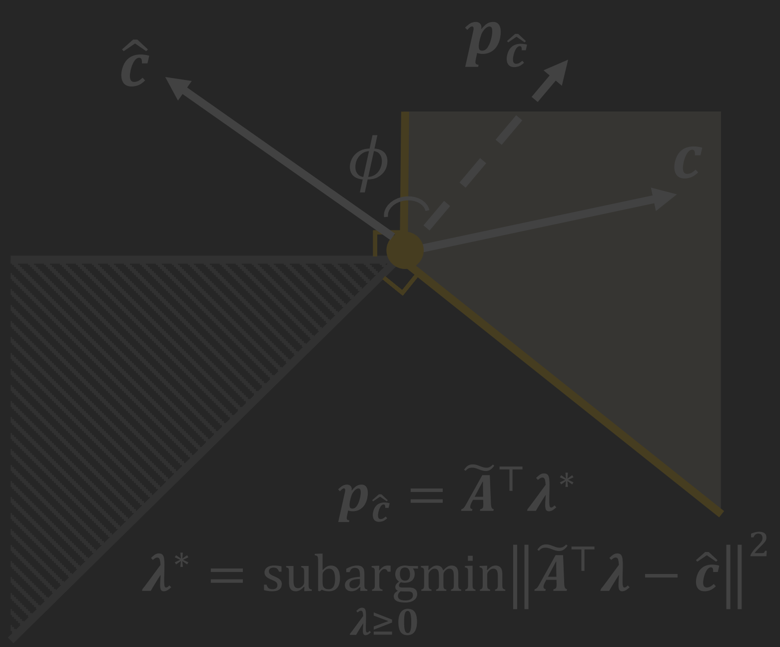

Linear Relaxation

# Projection



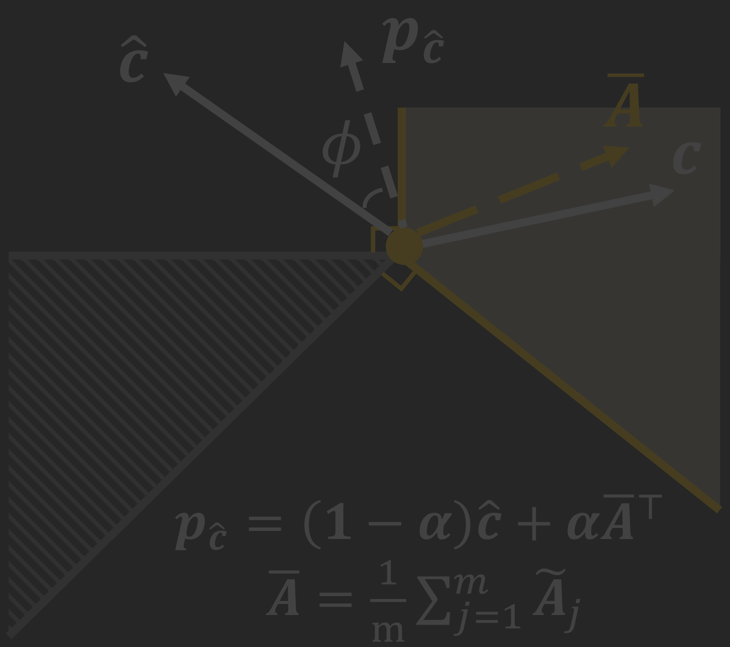

$$p_{\hat{c}} = \widetilde{A}^{\top}\lambda^{*}$$
$$\lambda^{*} = \underset{\lambda \geq 0}{\operatorname{argmin}} \left\| \widetilde{A}^{\top}\lambda - \hat{c} \right\|^{2}$$

**Exact Projection**




$$p_{\hat{c}} = \widetilde{A}^{\top}\lambda^{*}$$
$$\lambda^{*} = \underset{\lambda \geq 0}{\operatorname{subargmin}} \left\| \widetilde{A}^{\top}\lambda - \hat{c} \right\|^{2}$$

**Inner Projection**

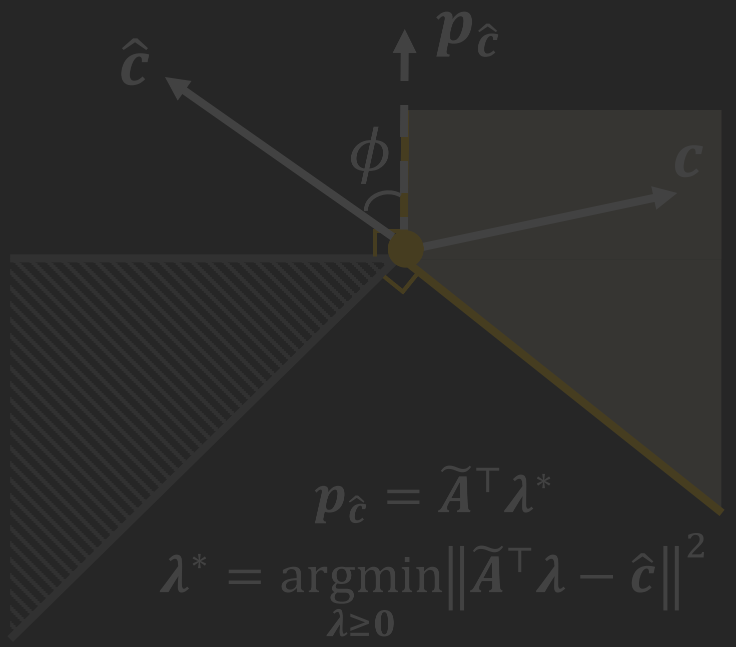



$$p_{\hat{c}} = (1-\alpha)\hat{c} + \alpha \bar{A}^{\top}$$
$$\bar{A} = \frac{1}{m}\sum_{j=1}^{m} \widetilde{A}_j$$

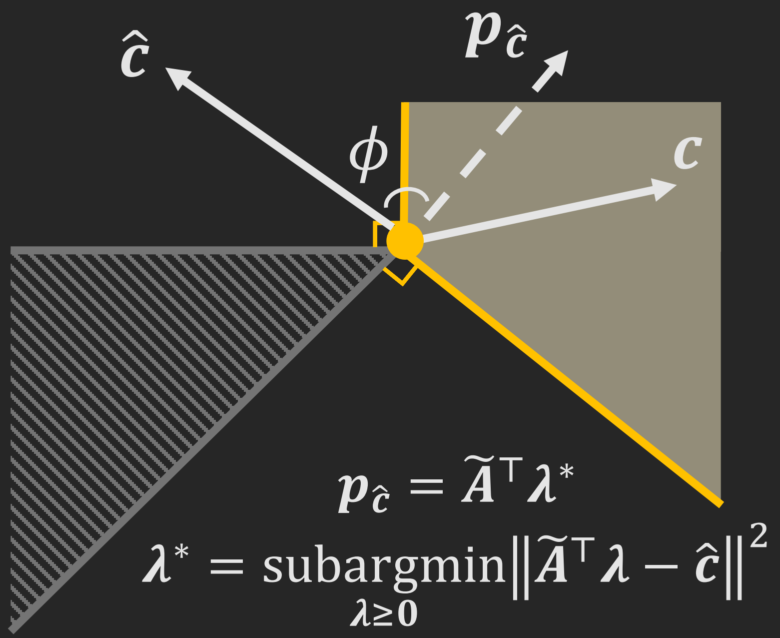
**Heuristic Projection**

CaVE Exact performs exact projection, wherein the optimal solution of the NNLS is on the surface of the cone.
This approach results in the vanishing gradients as the predicted cost vector close the surface.

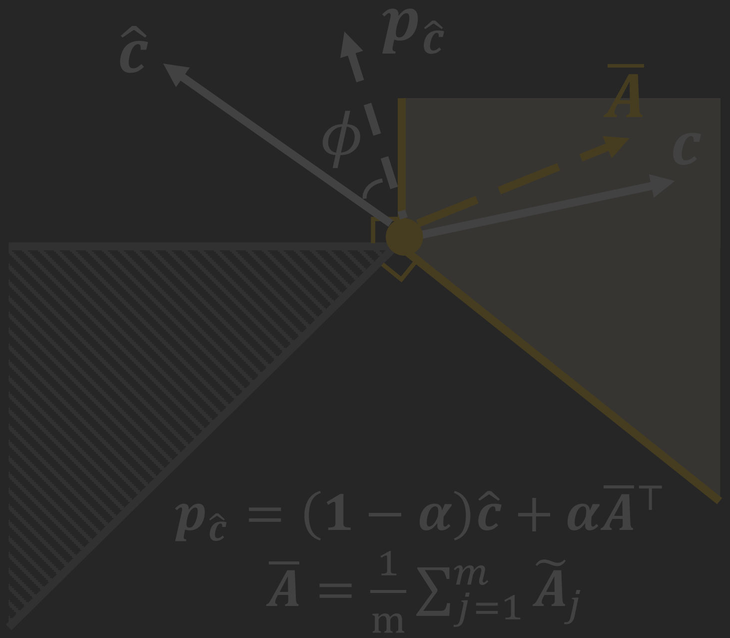
# Projection




Exact Projection




Inner Projection

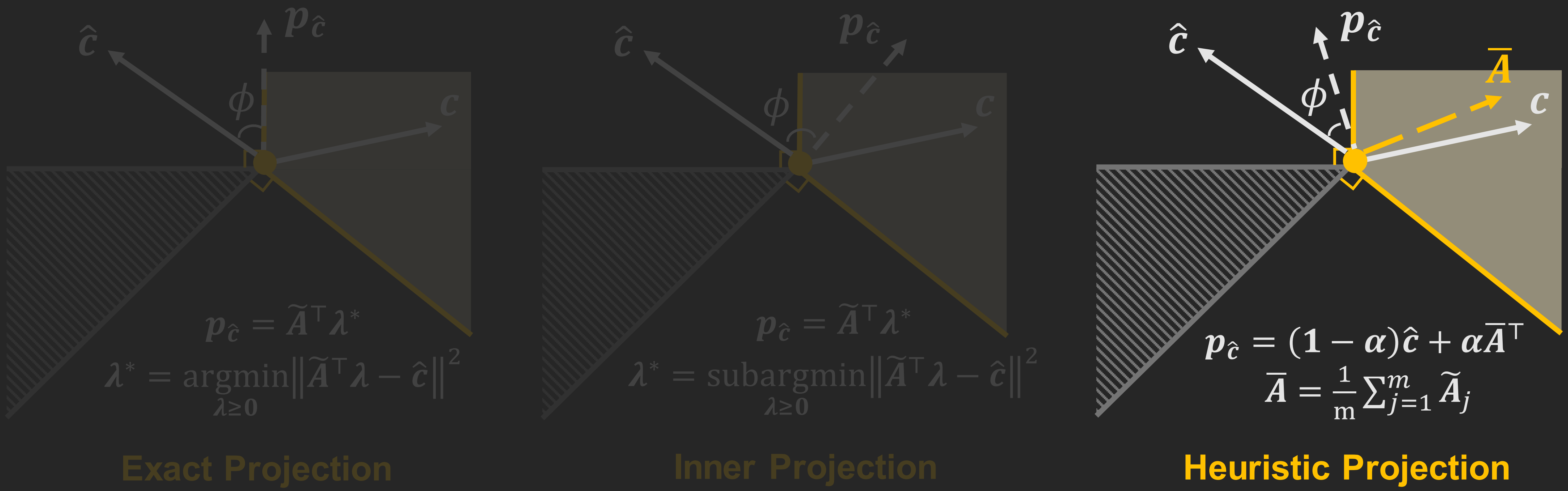



Heuristic Projection

The goal is to obtain a projection of the predicted cost vector that lies inside the subcone. (suboptimal solution)

Since the solver (Clarabel) uses the primal-dual interior point, the feasibility is guaranteed at each iteration. We can simply set the maximum iterations.

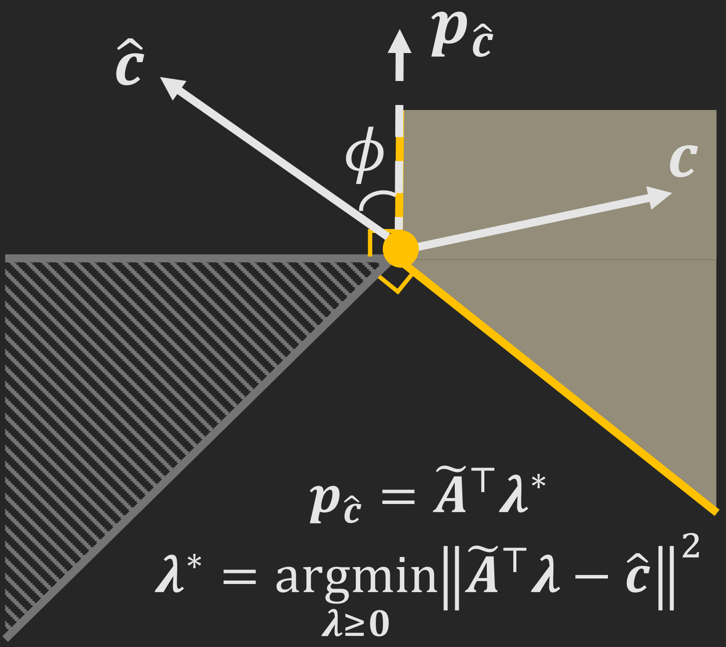
# Projection




A heuristic projection that does not require solving NNLS and relies on simple operations.

This approach does NOT guarantee feasibility, yet it ensures that the cost vector is pushed towards the cone.

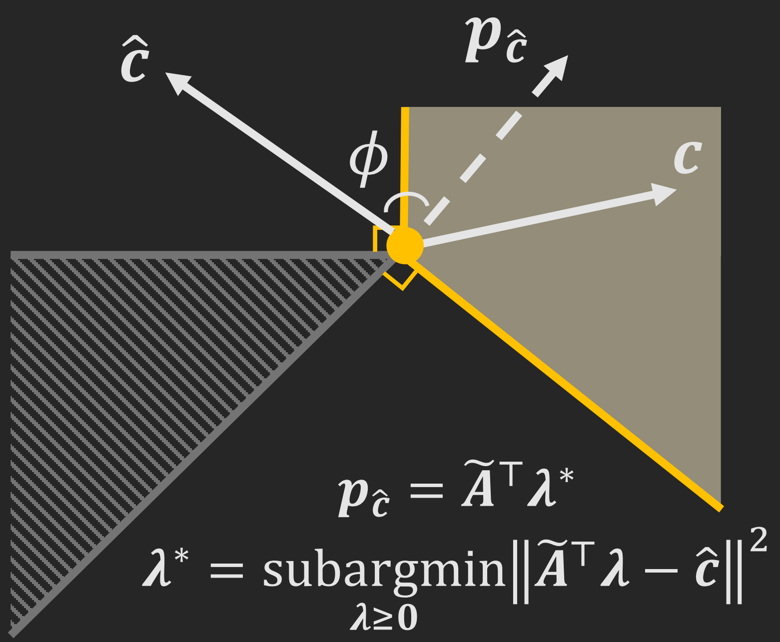
# Projection




$$p_{\hat{c}} = \widetilde{A}^\top \lambda^*$$
$$\lambda^* = \operatorname*{argmin}_{\lambda \geq 0} \left\| \widetilde{A}^\top \lambda - \hat{c} \right\|^2$$

**Exact Projection**

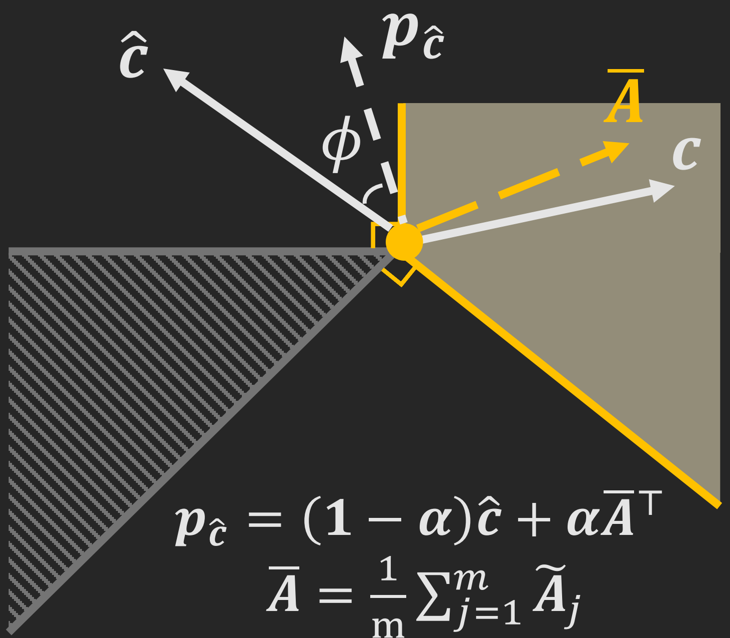
**CaVE Exact**




$$p_{\hat{c}} = \widetilde{A}^\top \lambda^*$$
$$\lambda^* = \operatorname*{subargmin}_{\lambda \geq 0} \left\| \widetilde{A}^\top \lambda - \hat{c} \right\|^2$$

**Inner Projection**

**CaVE+**




$$p_{\hat{c}} = (1 - \alpha)\hat{c} + \alpha \bar{A}^\top$$
$$\bar{A} = \frac{1}{m} \sum_{j=1}^{m} \widetilde{A}_j$$

**Heuristic Projection**

**CaVE Hybrid**

# Algorithm

**Algorithm** Cone-aligned Vector Estimation (`CaVE`)

**Require:** Pairs of feature vectors and binding constraints $\{(\boldsymbol{x}^i, \widetilde{\boldsymbol{A}}^i)\}_{i=1}^n$ for $n$ training instances; learning rate $\alpha > 0$

```
Initialize model parameters θ
for each training epoch do
    for each batch of training samples (x, Ã) do
        Predict cost coefficient ĉ ← g(x, θ)
        Compute projection p_ĉ with quadratic program
        Compute cosine similarity loss L_CaVE(ĉ, Ã)
        Compute the gradient ∇_θ L_CaVE(ĉ, Ã) with backpropagation
        Update ML model parameters θ ← θ − α∇_θ L_CaVE(ĉ, Ã)
    end for
end for
return g(·, θ)
```

# Experiments - SP5

## Shortest Path on 5 × 5 Grid

Average Test Normalized Regret (%) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 8.82 ± 1.15 | 10.73 ± 1.54 | 8.39 ± 0.95 | 8.35 ± 0.88 | 7.79 ± 1.00 | 7.68 ± 0.99 | 11.34 ± 1.11 |
| **Deg 6** | 12.58 ± 2.14 | 11.30 ± 1.30 | 8.89 ± 0.90 | 8.84 ± 1.00 | 7.72 ± 1.11 | 7.86 ± 0.96 | 13.78 ± 1.58 |

The degree of the polynomial from features to costs increases the complexity.
The higher the degree, the greater the difficulty.

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 1.52 ± 0.14 | 4.64 ± 0.09 | 4.89 ± 0.12 | 2.57 ± 0.19 | 17.64 ± 0.12 | 18.52 ± 0.31 | 4.50 ± 0.48 |
| **Deg 6** | 1.38 ± 0.13 | 3.52 ± 0.11 | 3.72 ± 0.14 | 2.39 ± 0.19 | 18.68 ± 0.40 | 17.78 ± 0.13 | 4.38 ± 0.42 |

# Experiments - SP5

## Shortest Path on 5 × 5 Grid

Average Test Normalized Regret (%) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 8.82 ± 1.15 | 10.73 ± 1.54 | 8.39 ± 0.95 | 8.35 ± 0.88 | 7.79 ± 1.00 | **7.68** ± 0.99 | 11.34 ± 1.11 |
| **Deg 6** | 12.58 ± 2.14 | 11.30 ± 1.30 | 8.89 ± 0.90 | 8.84 ± 1.00 | **7.72** ± 1.11 | 7.86 ± 0.96 | 13.78 ± 1.58 |

Average Training Time (Sec) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 1.52 ± 0.14 | 4.64 ± 0.09 | 4.89 ± 0.12 | **2.57** ± 0.19 | 17.64 ± 0.12 | 18.52 ± 0.31 | 4.50 ± 0.48 |
| **Deg 6** | 1.38 ± 0.13 | 3.52 ± 0.11 | 3.72 ± 0.14 | **2.39** ± 0.19 | 18.68 ± 0.40 | 17.78 ± 0.13 | 4.38 ± 0.42 |

# Experiments – TSP20

## Traveling Salesperson with 20 Nodes

### Average Test Normalized Regret (%) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 12.12 ± 0.89 | 7.35 ± 0.40 | 6.20 ± 0.24 | 7.69 ± 0.33 | **5.95** ± 0.16 | 6.56 ± 0.21 | 12.21 ± 0.88 |
| **Deg 6** | 21.32 ± 1.81 | 8.01 ± 0.45 | **6.97** ± 0.37 | 9.52 ± 0.64 | 7.48 ± 0.36 | 7.41 ± 0.37 | 14.31 ± 0.40 |

### Average Training Time (Sec) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 1.52 ± 0.10 | 113.56 ± 3.16 | 107.15 ± 3.80 | 27.06 ± 2.17 | 175.23 ± 4.95 | 220.21 ± 24.20 | **25.92** ± 4.23 |
| **Deg 6** | 1.53 ± 0.19 | 158.66 ± 9.65 | 102.19 ± 10.38 | 30.17 ± 2.62 | 185.13 ± 7.44 | 185.02 ± 5.09 | **25.48** ± 3.66 |

# Experiments – TSP50

## Traveling Salesperson with 50 Nodes

Average Test Normalized Regret (%) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 28.16 ± 1.08 | 15.19 ± 0.65 | 7.69 ± 0.22 | 9.59 ± 0.44 | **7.57** ± 0.20 | 8.03 ± 0.23 | 14.31 ± 0.40 |
| **Deg 6** | 52.61 ± 2.36 | 23.25 ± 2.41 | **8.57** ± 0.38 | 11.28 ± 0.80 | 10.26 ± 0.46 | 9.00 ± 0.52 | 17.12 ± 0.48 |

Average Training Time (Sec) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 1.55 ± 0.18 | 611.47 ± 23.52 | 518.07 ± 51.89 | 196.96 ± 35.92 | 1220.68 ± 85.39 | 1328.99 ± 28.87 | **151.80** ± 24.21 |
| **Deg 6** | 1.16 ± 0.13 | 502.71 ± 16.03 | 573.87 ± 20.19 | 253.93 ± 27.67 | 1191.29 ± 42.63 | 1456.21 ± 34.18 | **155.95** ± 24.46 |

# Experiments – CVRP20

## Capacity Vehicle Routing with 20 Nodes

Average Test Normalized Regret (%) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 10.10 ± 0.64 | 9.26 ± 1.56 | 6.44 ± 0.24 | 7.92 ± 0.52 | **5.94** ± 0.25 | 6.32 ± 0.28 | 15.77 ± 0.96 |
| **Deg 6** | 19.50 ± 1.22 | 11.64 ± 0.25 | **7.94** ± 0.54 | 11.44 ± 1.14 | 8.75 ± 0.28 | 8.09 ± 0.57 | 18.96 ± 1.01 |

Average Training Time (Sec) with Standard Deviation

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 1.65 ± 0.48 | 213.56 ± 42.36 | 153.56 ± 11.08 | **44.52** ± 6.27 | 7020.11 ± 1043.05 | 3773.31 ± 288.84 | 583.56 ± 170.67 |
| **Deg 6** | 1.54 ± 0.25 | 208.95 ± 12.90 | 127.94 ± 13.84 | **51.83** ± 8.78 | 2204.83 ± 99.86 | 6197.84 ± 288.63 | 470.20 ± 84.46 |

# Experiments – CVRP30

Due to the scale of the problem, we did not repeat our experimental evaluation with random seeds.

## Capacity Vehicle Routing with 30 Nodes

### Test Normalized Regret (%)

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 19.72 | 12.54 | **9.13** | 9.99 | N/A | | 18.28 |

### Training Time (Sec)

| Methods | 2-Stage | CaVE-E | CaVE+ | CaVE-H | SPO+ | PFYL | NCE |
|---|---|---|---|---|---|---|---|
| **Deg 4** | 9.27 | 331.73 | 287.77 | **132.62** | ≥100h | | 884.95 |

# Experiments – Relaxation

## Traveling Salesperson with 50 Nodes

Average Test Normalized Regret (%)

| Methods | CaVE+ | SPO+ Rel | PFYL Rel |
|---|---|---|---|
| Deg 4 | **7.69** ± 0.22 | 10.17 ± 0.23 | 11.11 ± 0.33 |
| Deg 6 | **8.57** ± 0.38 | 13.14 ± 0.46 | 13.38 ± 0.58 |

Average Training Time (Sec)

| Methods | CaVE+ | SPO+ Rel | PFYL Rel |
|---|---|---|---|
| Deg 4 | 518.07 ± 51.89 | **386.06** ± 9.69 | 536.67 ± 4.94 |
| Deg 6 | 573.87 ± 20.19 | 636.99 ± 3.04 | **510.37** ± 3.46 |

## Capacity Vehicle Routing with 20 Nodes

Average Test Normalized Regret (%)

| Methods | CaVE+ | SPO+ Rel | PFYL Rel |
|---|---|---|---|
| Deg 4 | **6.44** ± 0.24 | 8.03 ± 0.38 | 17.07 ± 0.63 |
| Deg 6 | **7.94** ± 0.54 | 15.73 ± 0.39 | 19.19 ± 1.66 |

Average Training Time (Sec)

| Methods | CaVE+ | SPO+ Rel | PFYL Rel |
|---|---|---|---|
| Deg 4 | 153.56 ± 11.08 | 78.95 ± 0.73 | **78.80** ± 1.19 |
| Deg 6 | 127.94 ± 13.84 | **78.74** ± 3.82 | 81.80 ± 0.86 |

# Thank You

GitHub