Explanations in CP

Satisfaction problems

- “Why has variable x value a in a/the solution?”
- “Why can variable x not have value b?”
- “Why is this problem UNSAT?”
- “Is there a solution where x = b?”
- “Are there any other solutions with other values for x?”
- …

Optimization problems

- “Why is x = a in a/the optimal solution?”
- “Why is an assignment x = b not optimal?”
- “Why is the objective function not higher/lower?”
- “How should the objective change so that x = b is optimal?”
- “What is the next best solution?”
- …

Explanations in CP

Satisfaction problems

- <u>“Why has variable x value a in a/the solution?”</u>
- “Why can variable x not have value b?”
- <u>“Why is this problem UNSAT?”</u>
- “Is there a solution where x = b?”
- “Are there any other solutions with other values for x?”
- …

Optimization problems

- “Why is x = a in a/the optimal solution?”
- “Why is an assignment x = b not optimal?”
- “Why is the objective function not higher/lower?”
- “How should the objective change so that x = b is optimal?”
- “What is the next best solution?”
- …

Explanations in CP

Satisfaction problems

- <u>“Why has variable x value a in a/the solution?”</u>
- “Why can variable x not have value b?”
- <u>“Why is this problem UNSAT?”</u>
- “Is there a solution where x = b?”
- “Are there any other solutions with other values for x?”
- …

Optimization problems

- “Why is x = a in a/the optimal solution?”
- “Why is an assignment x = b not optimal?”
- “Why is the objective function not higher/lower?”
- <u>“How should the objective change so that x = b is optimal?”</u>
- “What is the next best solution?”
- …

Counterfactual explanations

Given a constraint optimization problem with optimal solution x^*

<u>User:</u> “Why not the solution $\hat{x}$ I thought of instead of optimal x^*?”

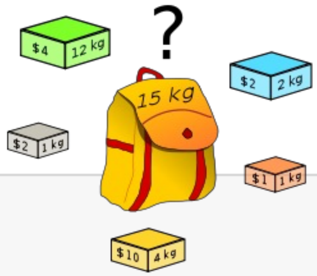
<u>Program:</u> “For $\hat{x}$ to be optimal the objective coefficients must change to d^*”

Ideally: changes to objective are as small as possible to ensure interpretability

Counterfactual explanations

Example

Knapsack:




```
model = Model()

gr,bl,og,ye,gy = boolvar(shape=5)

model += (12*gr + 2*bl + 1*og + 4*ye + 1*gy <= 15)

model.maximize(4*gr + 2*bl + 1*og + 10*ye + 2*gy)


model.solve()
```

```
print(gr.value(), bl.value(), og.value(), ye.value(), gy.value())
```

```
0 1 1 1 1
```

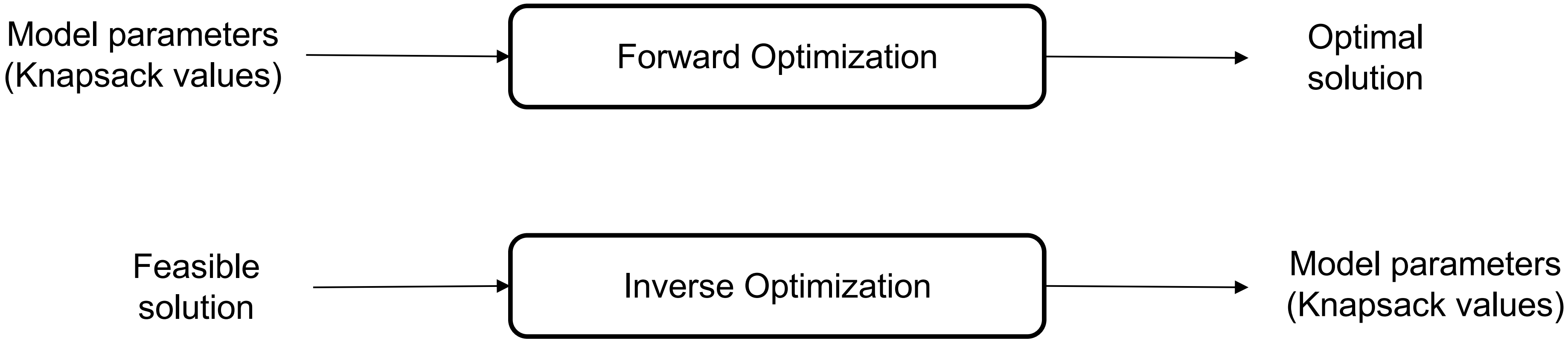
<u>User:</u>

“Why does the optimal not include gr?”

<u>Program:</u>

“Because to include gr into the knapsack, its value should change to at least 11 instead of 4”

(Inverse) Optimization



Inverse optimization and Counterfactual expl

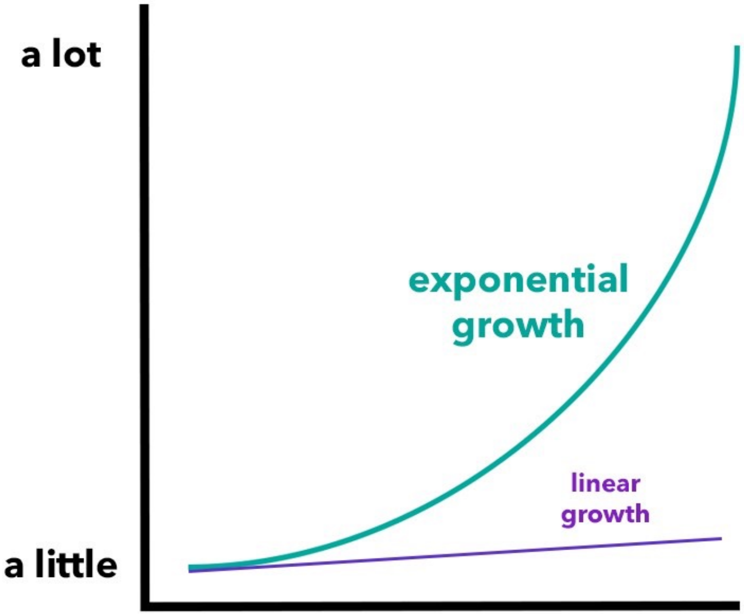
- User provides *part of* new solution $\hat{x}$
  - We want an explanation in terms of variables in this partial assignment
  - First extend partial solution to full solution [1]
- Find optimal objective for $\hat{x}$ and return as explanation

[1] Korikov, Anton, and J. Christopher Beck. "Counterfactual explanations via inverse constraint programming." *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

Inverse optimization

$$\begin{aligned}\mathbf{IO}(\mathbf{c}^0, \hat{\mathbf{x}}, \mathcal{X}): \ \underset{\mathbf{c}\in\mathcal{P}}{\text{minimize}} \quad & \left\|\mathbf{c} - \mathbf{c}^0\right\|_1 \\ \text{subject to} \quad & \mathbf{c}^\top \hat{\mathbf{x}} \le \mathbf{c}^\top \mathbf{x}_j, \quad \forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X})\end{aligned}$$

Inverse optimization



$$\forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X})$$

Finding cutting planes

- Master problem:
  - Add cut to feasible region and find new optimal cost vector

$$\begin{aligned} \mathbf{IO}(\mathbf{c}^0, \hat{\mathbf{x}}, \mathcal{X}): \ \underset{\mathbf{c}\in\mathcal{P}}{\text{minimize}} \quad & \left\|\mathbf{c} - \mathbf{c}^0\right\|_1 \\ \text{subject to} \quad & \mathbf{c}^\top \hat{\mathbf{x}} \le \mathbf{c}^\top \mathbf{x}_j, \quad \forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X}) \end{aligned}$$

- Sub problem:
  - Find extreme optimal point on convex hull given a cost vector c
  - I.e., solve the original forward problem with a new cost vector

$$\begin{aligned} \mathbf{FP}(\mathbf{c}, \mathcal{X}): \ \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} := \{\mathbf{A}\mathbf{x} \ge \mathbf{b}, \ \mathbf{x} \in \mathbb{Z}^{n-q} \times \mathbb{R}^q\}. \end{aligned}$$

Finding cutting planes

Iteratively add cuts → Incremental MIP solver

- Master problem:
  - Add cut to feasible region and find new optimal cost vector

$$\boxed{\begin{aligned} \mathbf{IO}(\mathbf{c}^0, \hat{\mathbf{x}}, \mathcal{X}): \underset{\mathbf{c}\in\mathcal{P}}{\text{minimize}} \quad & \left\|\mathbf{c} - \mathbf{c}^0\right\|_1 \\ \text{subject to} \quad & \mathbf{c}^\top \hat{\mathbf{x}} \le \mathbf{c}^\top \mathbf{x}_j, \quad \forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X}) \end{aligned}}$$

- Sub problem:
  - Find extreme optimal point on convex hull given a cost vector c
  - I.e., solve the original forward problem with a new cost vector

$$\begin{aligned} \mathbf{FP}(\mathbf{c}, \mathcal{X}): \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} := \{\mathbf{A}\mathbf{x} \ge \mathbf{b},\ \mathbf{x} \in \mathbb{Z}^{n-q} \times \mathbb{R}^q\}. \end{aligned}$$

Finding cutting planes

- Master problem:
  - Add cut to feasible region and find new optimal cost vector

Iteratively add cuts → Incremental MIP solver

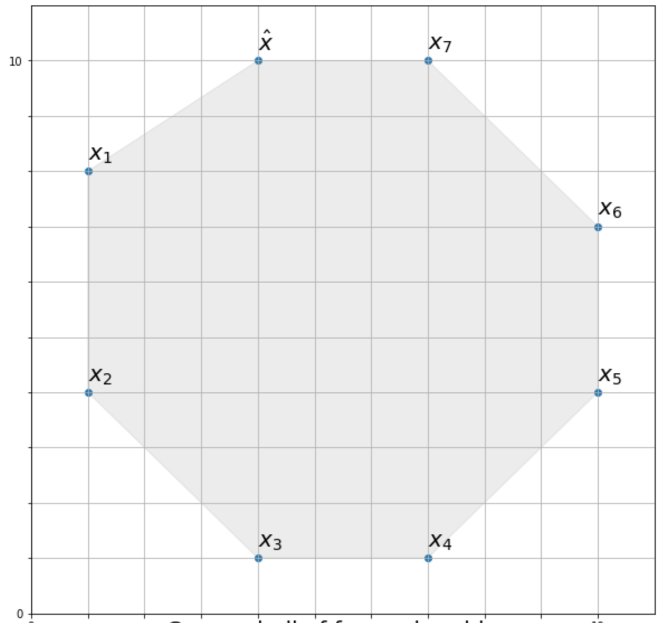
$$\boxed{\begin{aligned} \mathbf{IO}(\mathbf{c}^0, \hat{\mathbf{x}}, \mathcal{X}): \underset{\mathbf{c} \in \mathcal{P}}{\text{minimize}} \quad & \left\| \mathbf{c} - \mathbf{c}^0 \right\|_1 \\ \text{subject to} \quad & \mathbf{c}^\top \hat{\mathbf{x}} \leq \mathbf{c}^\top \mathbf{x}_j, \quad \forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X}) \end{aligned}}$$

- Sub problem:
  - Find extreme optimal point on convex hull given a cost vector c
  - I.e., solve the original forward problem with a new cost vector

$$\boxed{\begin{aligned} \mathbf{FP}(\mathbf{c}, \mathcal{X}): \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} := \{\mathbf{A}\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \mathbb{Z}^{n-q} \times \mathbb{R}^q\}. \end{aligned}}$$

Original problem → problem specific solver

Cutting planes for Inverse Optimization

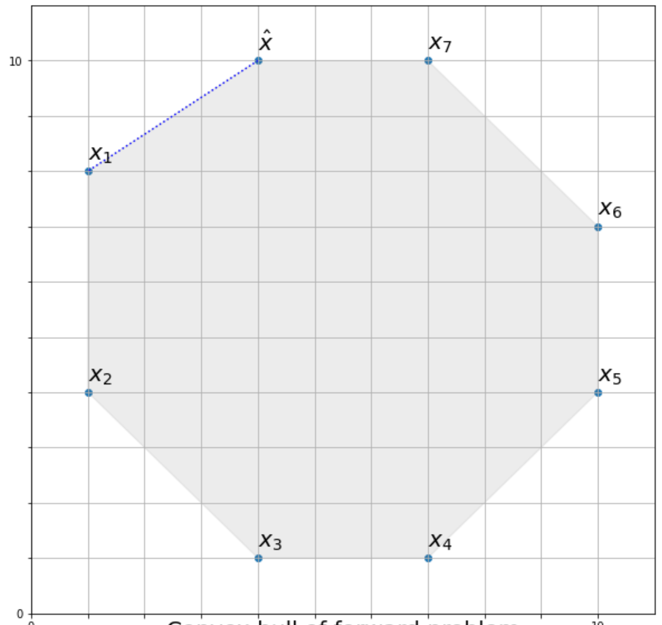


Convex hull of forward problem

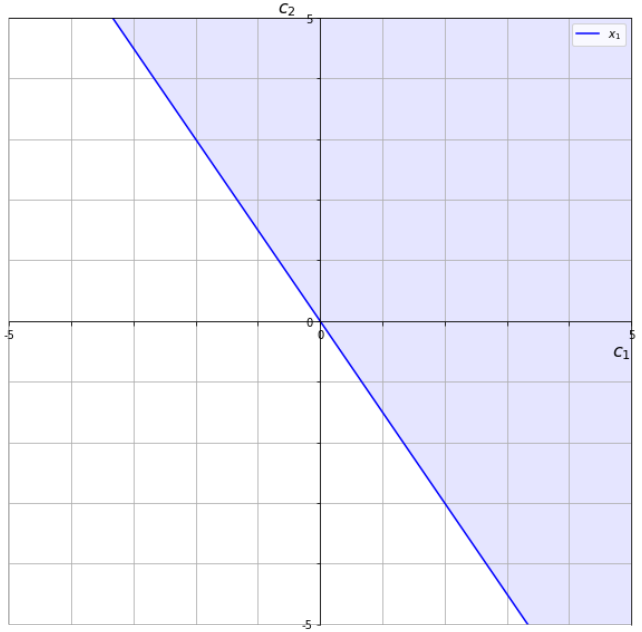
Every extreme point on the convex hull corresponds to an optimal solution for <u>*some*</u> cost vector c

Task: find the cost vector <u>making $\hat{x}$ optimal</u>
while minimizing changes to origninal cost vector

Cutting planes for Inverse Optimization



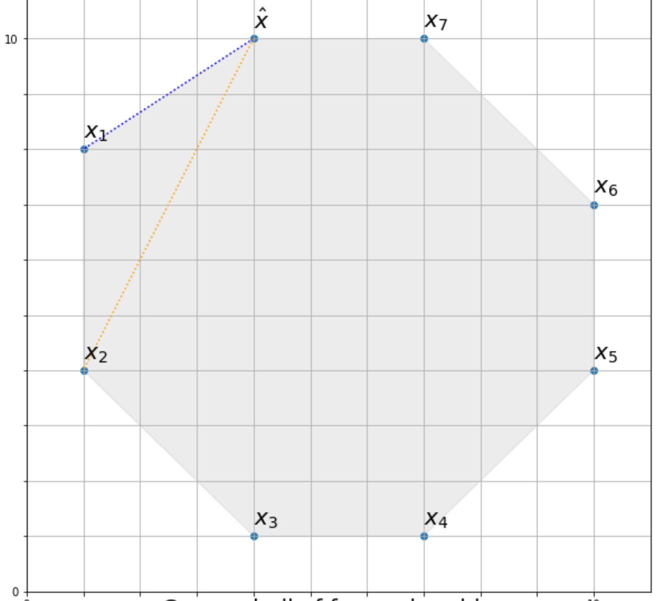
Convex hull of forward problem



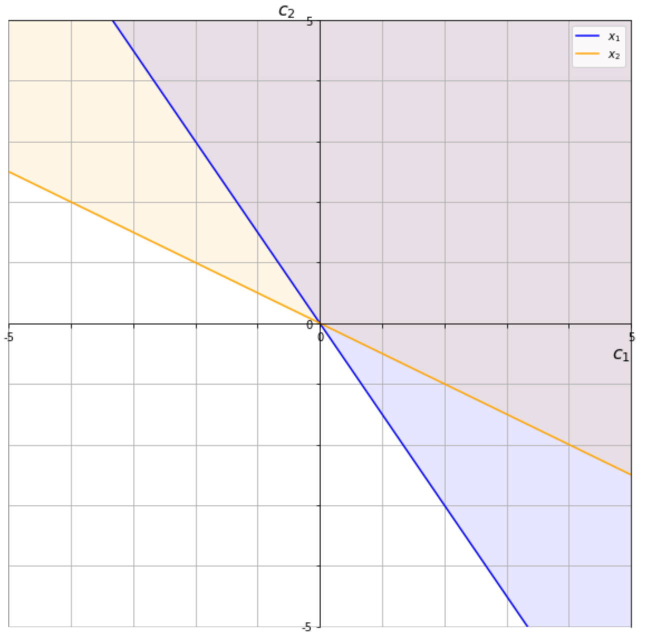
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



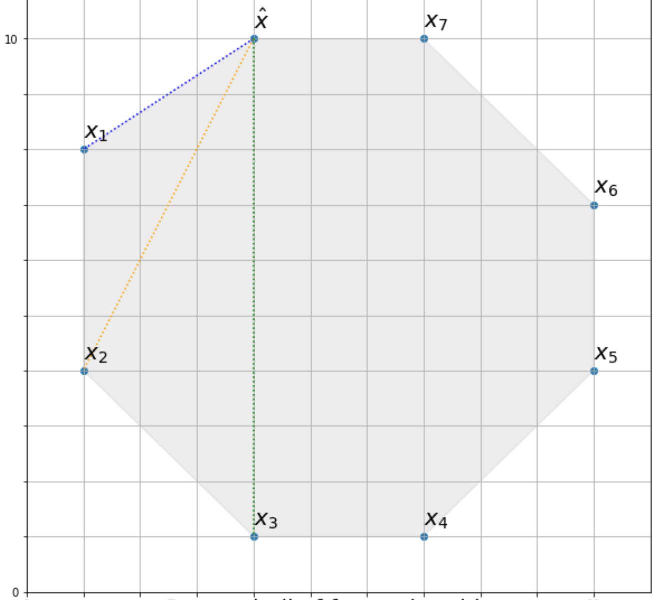
Convex hull of forward problem



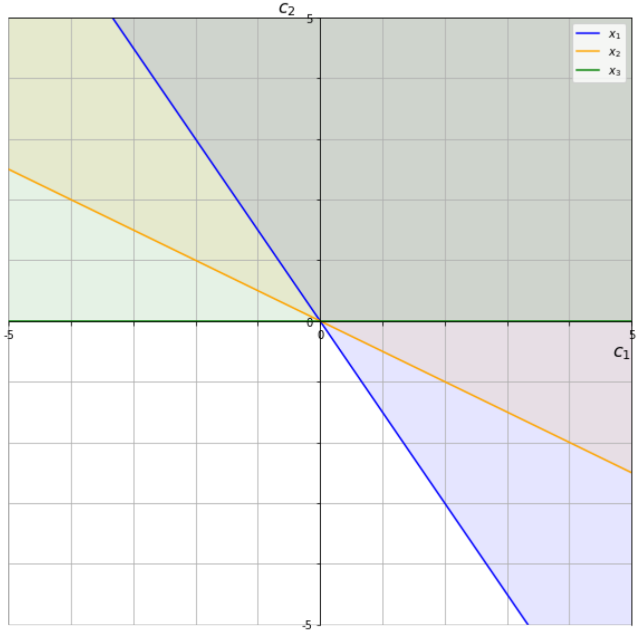
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



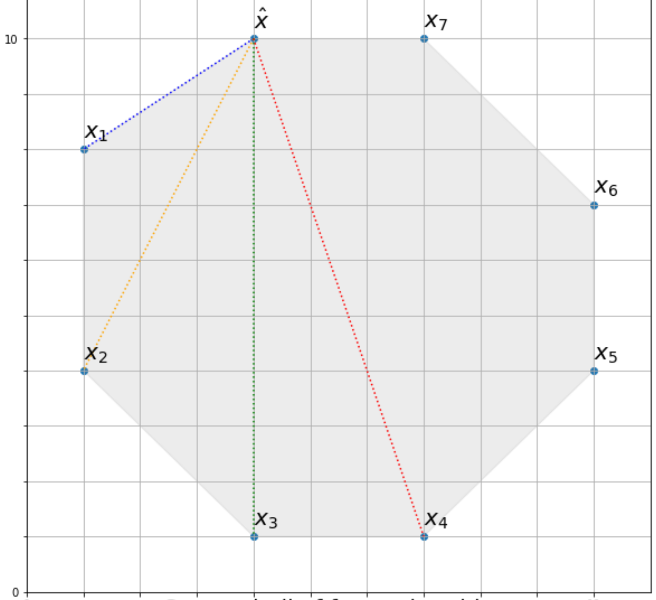
Convex hull of forward problem



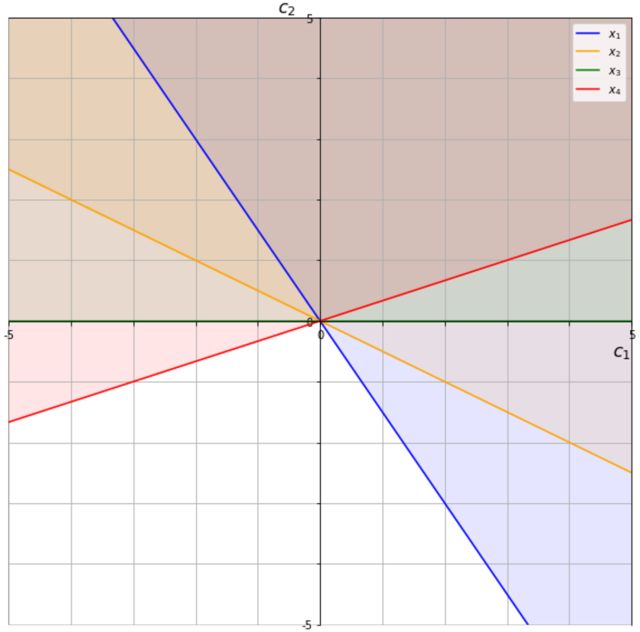
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



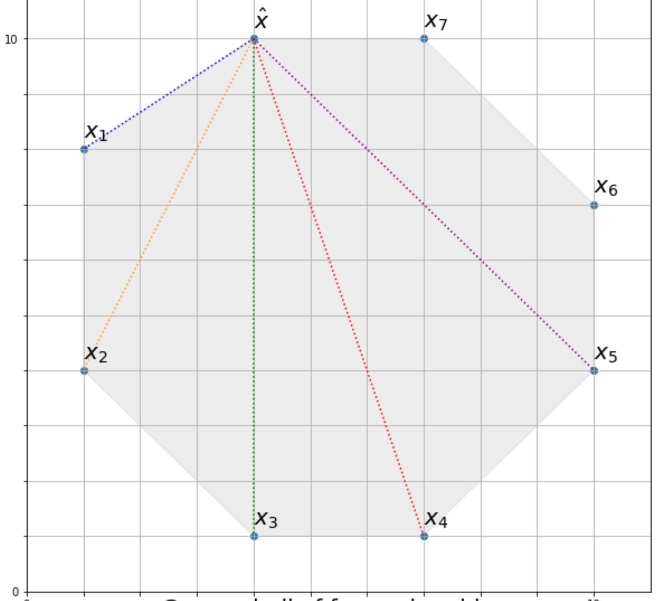
Convex hull of forward problem



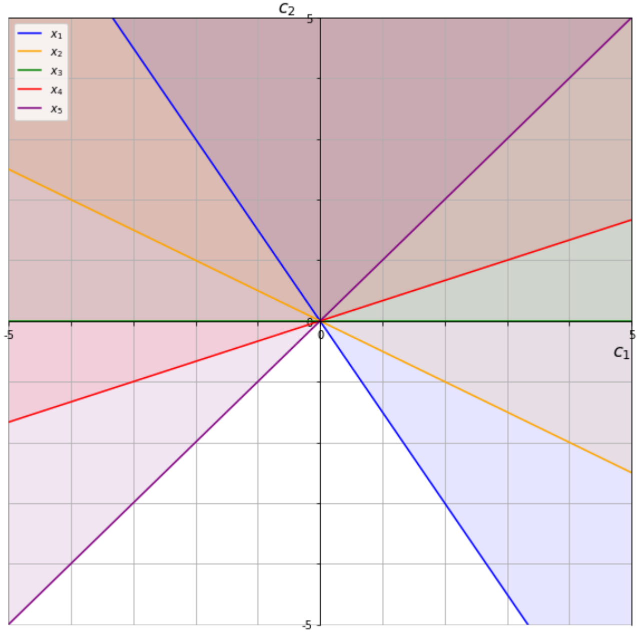
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



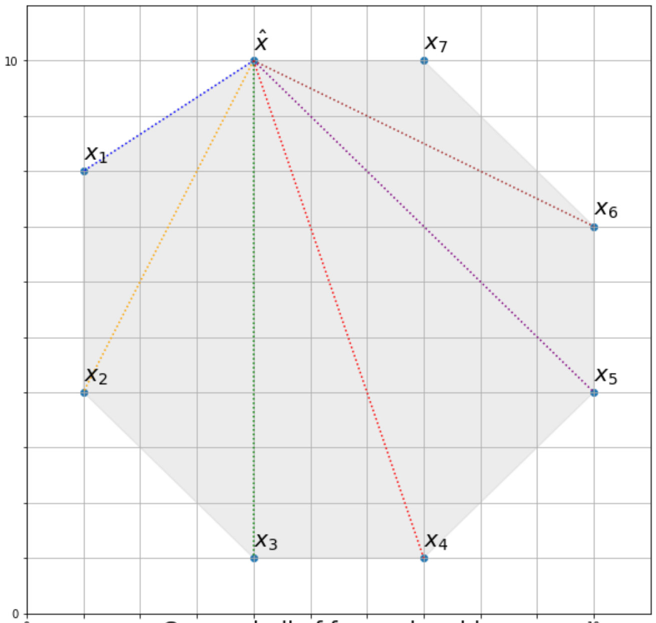
Convex hull of forward problem



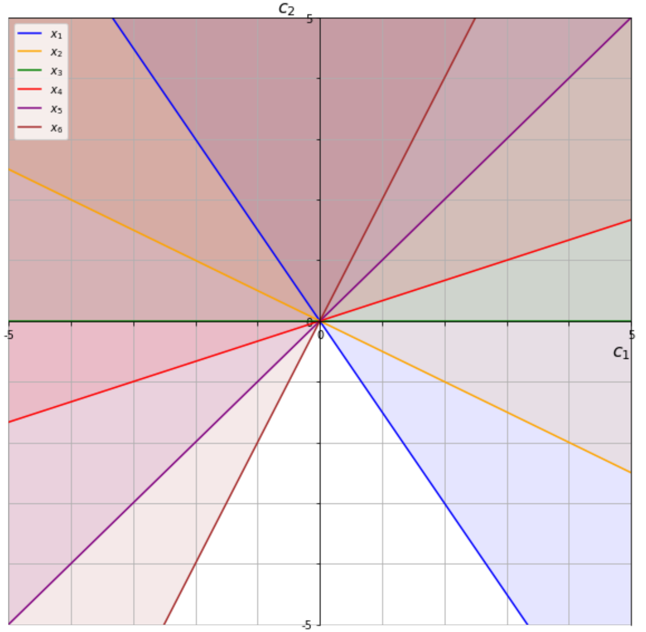
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



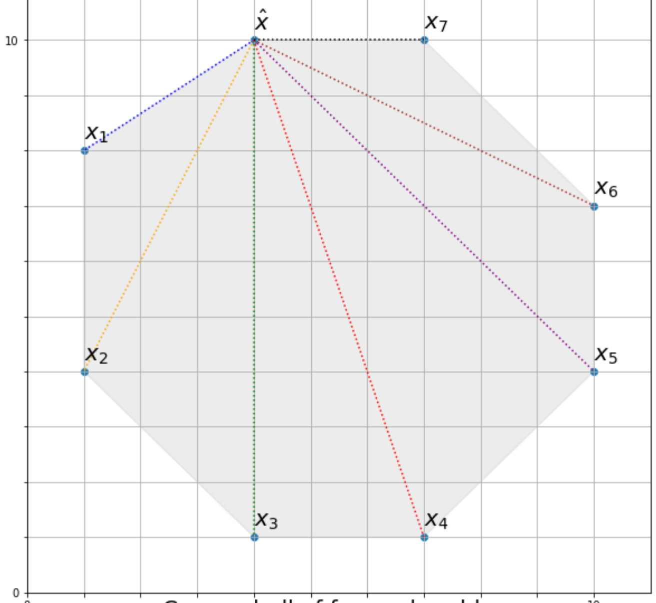
Convex hull of forward problem



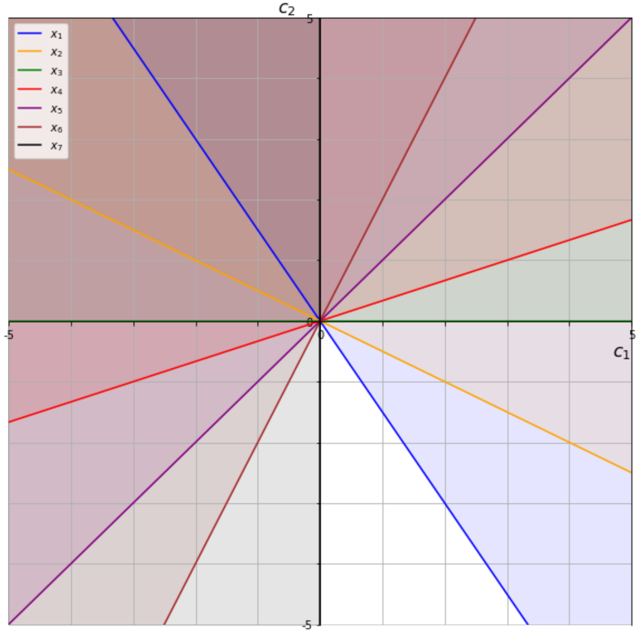
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



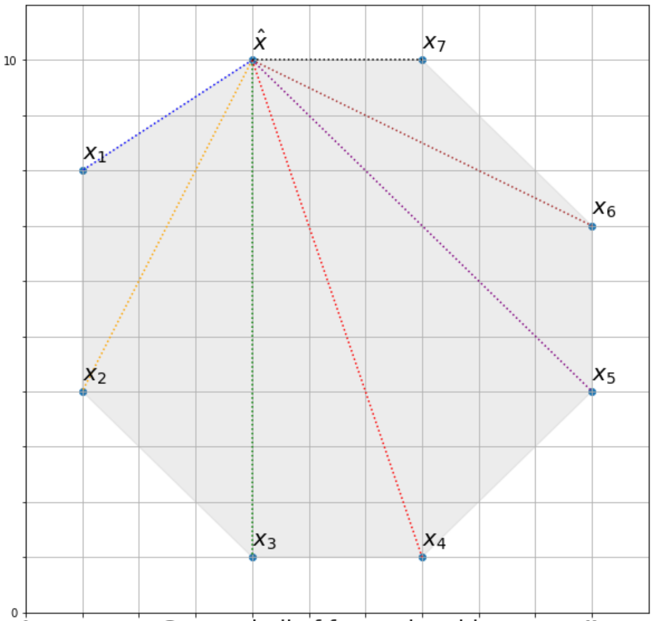
Convex hull of forward problem



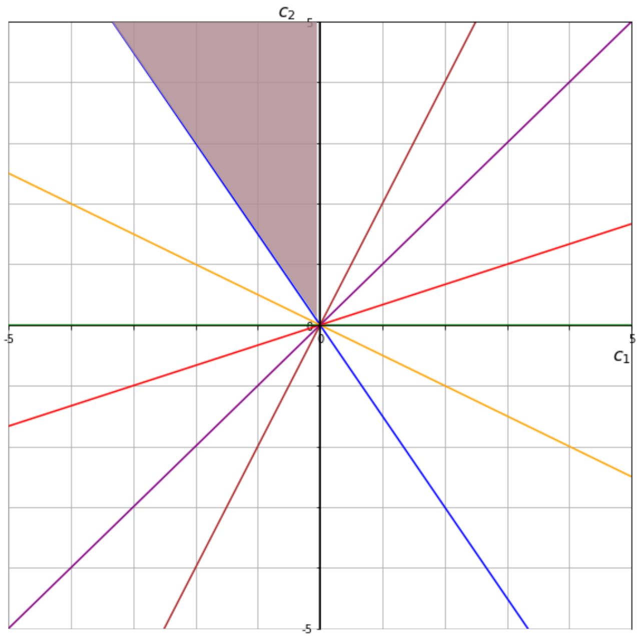
Half spaces induced by cutting planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Cutting planes for Inverse Optimization



Convex hull of forward problem



Intersection of all half planes

Bodur, Merve, Timothy CY Chan, and Ian Yihang Zhu. "Inverse Mixed Integer Optimization: Polyhedral Insights and Trust Region Methods." INFORMS Journal on Computing (2022).

Finding cutting planes

- Master problem:
  - Add cut to feasible region and find new optimal cost vector

$$\mathbf{IO}(\mathbf{c}^0, \hat{\mathbf{x}}, \mathcal{X}): \underset{\mathbf{c}\in\mathcal{P}}{\text{minimize}} \quad \left\|\mathbf{c} - \mathbf{c}^0\right\|_1$$
$$\text{subject to} \quad \mathbf{c}^\top \hat{\mathbf{x}} \leq \mathbf{c}^\top \mathbf{x}_j, \quad \forall \mathbf{x}_j \in \mathcal{E}(\mathcal{X})$$

- Sub problem:
  - Find extreme optimal point on convex hull given a cost vector c
  - I.e., solve the original forward problem with a new cost vector

$$\mathbf{FP}(\mathbf{c}, \mathcal{X}): \underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}^\top \mathbf{x}$$
$$\text{subject to} \quad \mathbf{x} \in \mathcal{X} := \{\mathbf{A}\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \mathbb{Z}^{n-q} \times \mathbb{R}^q\}.$$

Inverse optimization in CPMpy

```
def inverse_optimize(SP, c, x, x_d, keep_static=None):

    # Decision variable for new parameter vector
    d = intvar(0,INFTY, shape=len(x_d), name="d")

    # create the master problem
    MP = SolverLookup.get("gurobi")
    MP.minimize(norm(c-d,1))
    MP += SP.constraints
    while MP.solve():
        # find new cost vector
        new_d = d.value()
        print(f"New costvector = {new_d}")

        # find point on convex hull corresponding to new_d
        SP.maximize(sum(new_d * x))
        SP.solve()

        if sum(new_d * x_d) >= sum(new_d * x.value()):
            # solution is optimal
            break

        # add new cut to MP
        MP += sum(d * x_d) >= sum(d * x.value())

    return new_d, x.value()
```

Minimizing L1 norm

Solving the forward problem

Iteratively building cut constraints

Example: Knapsack problem

- Solver finds optimal solution to given problem:

```
model, (items, values, weights, capacity) = get_knapsack_problem()
assert model.solve()
print("Objective value:",model.objective_value())
print("Used capacity:", sum(items.value() * weights))

print(f"{values = }")
print(f"{weights = }")
print(f"{capacity = }")

print(f"items = {items.value()}")
```

```
Objective value: 32
Used capacity: 31
values = array([5, 0, 3, 3, 7, 9, 3, 5])
weights = array([2, 4, 7, 6, 8, 8, 1, 6])
capacity = 35
items = [ True False False  True  True  True  True  True]
```

Example: knapsack problem

```
items = [ True False False  True  True  True  True  True]
```

```
# User query
# "I want my solution to really contain item 1 and 2"
model += all(items[[1,2]])
assert model.solve()

x_hat = items.value()
print("Objective value:",model.objective_value())
print("Used capacity:", sum(x_hat * weights))

print(f"Items: {x_hat}")
```

```
Objective value: 29
Used capacity: 35
Items: [ True  True  True False  True  True False  True]
```

Find new solution satisfying extra constraints

Korikov, Anton, and J. Christopher Beck. "Counterfactual explanations via inverse constraint programming." *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

Example: knapsack problem

New objective only changes for items 1 and 2!

```
Original values: [5 0 3 3 7 9 3 5]
new costvector = [5 0 3 3 7 9 3 5] New cut = [ True False False  True  True  True  True  True]
new costvector = [5 0 6 3 7 9 3 5] New cut = [ True False  True False  True  True  True  True]
new costvector = [5 3 3 3 7 9 3 5] New cut = [ True  True False  True  True  True  True  True]
new costvector = [5 3 6 3 7 9 3 5] New cut = [ True  True False  True  True  True  True  True]

array([5, 3, 6, 3, 7, 9, 3, 5])
```

Demo

Counterfactual explanations

examples/tutorial_ijcai22/7_counterfactual_explain.ipynb

Explanations in CP

Satisfaction problems

- “Why has variable x value a in a/the solution?”
- “Why can variable x not have value b?”
- “Why is this problem UNSAT?”
- “Is there a solution where x = b?”
- “Are there any other solutions with other values for x?”
- …

Optimization problems

- “Why is x = a in a/the optimal solution?”
- “Why is an assignment x = b not optimal?”
- “Why is the objective function not higher/lower?”
- “How should the objective change so that x = b is optimal?”
- “What is the next best solution?”
- …

Conclusions

- Several applications in CP require repeated solver calls
  - Use CP/SAT/MIP-solvers as an oracle
  - Big efficieny gains using incremental solving
  - Use complementary strengths of several solvers

- Many applications in explanation generation:
  - MUS enumeration/extraction
  - OUS/SMUS Implicit hitting set algorithms
  - OCUS Implicit hitting set algorithms with constraints
  - Inverse optimization with cutting planes
  - …

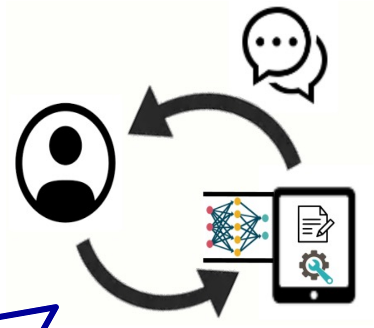
Conclusions

- CPMpy is the right tool for the job:
  - Easy prototyping in Python/Numpy
  - Supports many solver paradigms
  - Incremental solving
  - Open source
    - We welcome all contributions/feedback!

https://github.com/CPMpy/cpmpy

European Research Council
Established by the European Commission

CHAT-Opt:
Conversational **H**uman-**A**ware **T**echnology for **Opt**imisation



Towards **co-creation** of constrained optimisation solutions

- Solver that learns from user and environment
- Towards conversational: explanations and stateful interaction

https://people.cs.kuleuven.be/~tias.guns
@TiasGuns

Hiring post-docs!