Machine Learning for Two-stage Stochastic Programming

Yifeng Xiao

July 4, 2023

Content

- Intro
- Methods
  - Scenario reduction
    - Representation learning for scenarios
    - Objective value measurement of scenario similarity
  - Approximation
    - Objective value
    - Terms in traditional method

Intro

Mathematical model:

$$\begin{aligned} \min_{x} \quad & h(x) + \mathbb{E}_{\xi}[Q(x,\xi)], \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned}$$

where the x is the first-stage decision, and $Q(x,\xi)$ is the optimal value of second-stage programming.
Typical problems:
Capacitated Facility Location Problem, Investment Problem, Stochastic Server Location Problem, Pooling Problem.

Intro

General model:

A general stochastic programming may be like:

$$\min_{x,y} \quad \mathbb{E}_{\xi}[f(x,y;\xi)],$$

Letting $g(x) = min_y f(x,y;\xi)$, we have following equation:

$$\min_{x,y} \mathbb{E}_{\xi}[f(x,y;\xi)] = \min_{x} \mathbb{E}_{\xi}[g(x)] = \min_{x} \mathbb{E}_{\xi}[\min_{y} \ f(x,y;\xi)] \tag{1}$$

Intro

Example:Newsvendor

First stage: purchase quantity x, cost $c^\top x$
Second stage: selling quantity y, selling price p, salvage value v, random demand ξ

$$\begin{aligned} \min_{x,y} \quad & c^\top x + \mathbb{E}[Q(x,\xi)], \\ \text{s.t.} \quad & x \in \mathbb{Z}^+ \end{aligned}$$

where the second-stage problem is:

$$\begin{aligned} \min_{y} \quad & Q(x,\xi) = -py - v(x-y)^+, \\ \text{s.t.} \quad & y \in \mathbb{Z}^+, \\ & y \leq x, y \leq \xi. \end{aligned}$$

Intro

Difficulty:

The optimal value expectation of the second-stage programming

$$\mathbb{E}_{\xi}[Q(x, \xi)]$$

Expectation term results in problems in computation, which even worsens when the second-stage programming is non-linear.
Practically, we tend to **use discrete distribution to approximate the expectation**, because the real distribution of ξ is hard to know:

$$\mathbb{E}_{\xi}[Q(x, \xi)] \approx \sum_{i=1}^{N} p_{\xi_i} Q(x, \xi_i)$$

Intro

After sampling scenarios $\{\xi_i\}_{i=1}^N$, the problem becomes (extensive form):

$$
\begin{aligned}
\min_{x,y} \quad & h(x) + \sum_{i=1}^{N} p_i F[(y_i, \xi_i)] \\
\text{s.t.} \quad & x \in \mathcal{X}, \\
& y_i \in \mathcal{Y}(x, \xi_i) \quad i = 1, ..., N,
\end{aligned}
$$

where $\{\xi_i\}$ are N scenarios sampled from a probability distribution $\mathbb{P}$ (assumed), and we let $Q(x, \xi) = min_y\{F(y, \xi) : y \in \mathcal{Y}(x, \xi)\}$.

In linear case:
Objective function: $h(x) = c^\top x$, $F[(y_i, \xi_i)] = q_{\xi_i}^\top y_i$
Constraints: $\mathcal{X} = \{x : Ax \geq b\}$, $\mathcal{Y}_\xi = \{y_i : W_{\xi_i} y_i \geq h_{\xi_i} - T_{\xi_i}^\top x\}$

Method

Sampling usually results in the **explosion of problem size**, because the number of scenarios is large. So there are several kinds of ML-based methods for improvement:

- Scenario reduction
- Approximation

Scenario reduction

A natural improvement is to limit the number of scenarios N.

Scenarios $\{\xi_1, \ldots, \xi_N\}$ are sampled from a random vector that follows a distribution $\mathbb{P}$ (as a vector or an operator), and there may be some groups in $\{\xi_1, \ldots, \xi_N\}$.

$\Downarrow$

Scenario clustering

Picking one sample with greater weight in each group can be a choice to reduce the number N while keeping the samples representative.

Scenario reduction

The curse of dimensionality

The problems that arise when the dimensionality of data increases, including computational cost and sparsity.

- Distance computing: $\sqrt{\sum_{i=1}^{M}(x_i^2 - y_i^2)}$
- Sparsity: volume of n-dimensional unit ball $\to 0$ as $n \to \infty$

A Solution is dimensionality reduction $\Longleftrightarrow$ sample representation.

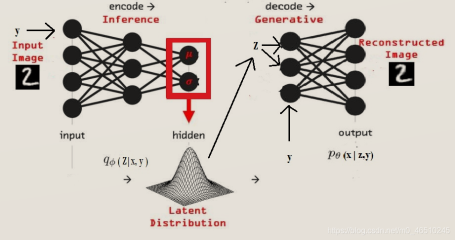
Scenario reduction

Learning Scenario Representation for Solving Two-stage Stochastic Integer Programs [WSCZ22]

Abstract

- Dimension reduction:
  Aim to learn a mapping from high-dimensional scenarios $\{w_i\}_{i=1}^N$ to low-dimensional latent representation $\{z_i\}_{i=1}^N$ under problem context D (static parameters in the first-stage programming) with CVAE (conditional variable auto-encoder).
- Clustering

CVAE



CVAE can be regarded as a Supervised version of VAE, with labels added into the inputs of the encoder and decoder.

Scenario representation learning model

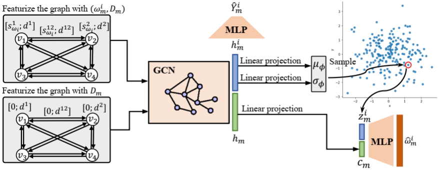


Figure 1: CVAE structure for learning scenario representations

- the context D does not donate label
- the input $\{w_i\}_{i=1}^N$ is compounded with the context D, which means the it is not only embedding of scenarios, but scenarios under contexts
- the encoder consists of a GCN (shared with decoder for embedding D) and following linear layers, the decoder consist of a MLP
- additional sub-network for prediction of objective values, **it acts like a downstream task to make the embedding more functional**.

Scenario reduction

Details in GCN

- The GCN and its input are **specifically designed based on the problem**.
  For example, in NDP, the nodes in the GCN are the nodes in the NDP. The input features of nodes $s^j_{w_i}$ denotes a realization of uncertain parameters on the j-th node. d^j denotes static parameters on the j-th node. It is the same when it comes to the edges.

	Feats	Types	Parameters	Digits
NDP	$\mathbf{v}^j_{\omega_i}$	d^j	None	0
		$s^j_{\omega_i}$	demand	1
	$\mathbf{e}^{jk}_{\omega_i}$	d^{jk}	opening cost, transportation cost, capacity	3
		$s^{jk}_{\omega_i}$	None	0
FLP	$\mathbf{v}^j_{\omega_i}$	d^j	coordinate, opening cost (***F.***), capacity (***F.***)	2(***4***)
		$s^j_{\omega_i}$	presence (***C.***)	1
	$\mathbf{e}^{jk}_{\omega_i}$	d_{jk}	distance between nodes	1
		$s^{jk}_{\omega_i}$	None	0

Figure: Specific details

Scenario reduction

Details in GCN

- Half-convolution is implemented on the GNN:

$$v_j^{l+1} = v_j^l + PReLU(BN(v_j^l W_1^l + \sum_{e_{jk}} \eta_{jk}^l \odot v_j^l W_2^l)), \eta_{jk}^l = \frac{\sigma(e_{jk}^l)}{\sum(\sigma(e_{jk}^l)) + \epsilon},$$

$$e_{jk}^{l+1} = e_{jk}^l + PReLU(BN(e_{jk}^l W_3^l + v_j^l W_4^l + v_k^l W_5^l)),$$

where $\odot$ donate element-wise product.

- The graph embedding is attained by **mean pooling over all nodes** while the edge embeddings are neglected.

Scenario reduction

Training

Loss function

$$\mathcal{L} = MSE(w, \hat{w}) + \beta KLdivergence(z, N(0,1)) \\ + \alpha MSE(y, groundtruth(\hat{y}))$$

where the w denotes a scenario, and the y denotes the output of sub-network, predicted optimal objective value. In the experiments, $\beta = 0.005$ and $\alpha = 100$.

Data

12800 instances (different problem sizes) and 200 scenarios for each one. The objective values are from CPLEX and account for 1% among total data because this part is hard to generate.

Scenario reduction

Experiments

	NDP (14; 200)									FLP (30; 200)								
	K=5			K=10			K=20			K=5			K=10			K=20		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	635.75	0.00	18s	635.75	0.00	18s	635.75	0.00	18s	236.71	0.00	18s	236.71	0.00	18s	236.71	0.00	18s
Scenario-M	2533.50	2.96	0.8s	2533.50	2.96	0.8s	2533.50	2.96	0.8s	1980.03	7.11	0.6s	1980.03	7.11	0.6s	1980.03	7.11	0.6s
Solution-M	798.81	0.26	0.8s	798.81	0.26	0.8s	798.81	0.26	0.8s	736.21	1.87	0.2s	736.21	1.87	0.2s	736.21	1.87	0.2s
K-medoids	976.47	0.54	0.3s	761.18	0.19	0.6s	677.02	0.08	1s	2390.06	9.06	0.2s	1344.39	4.57	0.3s	502.68	1.17	1s
CVAE-SIP	930.73	0.48	0.7s	734.24	0.15	0.9s	**637.99**	**0.02**	1s	929.61	2.87	0.7s	482.70	0.99	0.7s	282.49	0.23	1s
CVAE-SIPA	**769.70**	**0.24**	0.6s	**687.68**	**0.08**	0.8s	642.12	0.03	1s	**709.08**	**1.71**	0.6s	**381.41**	**0.58**	0.7s	**264.01**	**0.15**	1s

[1] $(n; N)$ means n nodes and N scenarios; **Bold** means the best result from the learning based methods.

When the training dataset is not big enough, using objective value to measure scenarios (additional downstream value-prediction task) can make the training more efficient, which demands that the embedding should be good enough to recover the objective value.

Scenario reduction

Experiments

Table 3: Generalization to large-scale problems

	NDP (24; 200)									FLP (60; 200)								
	K=5			K=10			K=20			K=5			K=10			K=20		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	602.47	0.00	2m	602.47	0.00	2m	602.47	0.00	2m	335.37	0.00	11m	335.37	0.00	11m	335.37	0.00	11m
Scenario-M	2238.63	2.68	2s	2238.63	2.68	2s	2238.63	2.68	2s	-	-	-	-	-	-	-	-	-
Solution-M	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
K-medoids	981.94	0.63	0.8s	723.62	0.20	1s	662.46	0.09	3s	3549.25	9.72	1s	1549.53	3.74	4s	561.97	0.73	7s
CVAE-SIP	871.73	0.45	2s	696.22	0.16	2s	659.92	0.09	4s	1083.12	2.17	3s	748.59	1.21	5s	506.75	0.57	8s
CVAE-SIPA	**809.65**	**0.34**	2s	**637.72**	**0.06**	2s	**610.58**	**0.01**	5s	**974.17**	**1.86**	3s	**746.02**	**1.18**	5s	**470.39**	**0.44**	8s

	NDP (44; 200)									FLP (120; 200)								
	K=5			K=10			K=20			K=5			K=10			K=20		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	580.67	0.00	23m	580.67	0.00	23m	580.67	0.00	23m	484.60	0.00	1h	484.60	0.00	1h	484.60	0.00	1h
Scenario-M	2163.74	2.77	3s	2163.74	2.77	3s	2163.74	2.77	3s	-	-	-	-	-	-	-	-	-
Solution-M	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
K-medoids	803.99	0.38	3s	652.78	0.12	7s	595.15	0.04	14s	6675.13	12.56	6s	2803.77	4.86	21s	1183.44	1.56	61s
CVAE-SIP	1079.80	0.84	5s	**615.99**	**0.06**	8s	588.18	0.02	16s	**1585.53**	**2.27**	12s	985.45	0.99	35s	627.30	0.28	56s
CVAE-SIPA	**682.35**	**0.17**	5s	627.62	0.08	8s	**581.07**	**0.00**	17s	2051.45	3.13	16s	**910.50**	**0.84**	33s	**594.33**	**0.22**	69s

Although the number of nodes can change between instances, the form feature vectors at nodes remains the same, which enables generalization between problem sizes.

Scenario reduction

Problem-driven scenario clustering in stochastic optimization [KOR23]

Abstract

Last paper, the objective values influence the scenario representation indirectly. This research regards the objective value as the only measurement to cluster scenarios, and aims to cluster scenarios by minimizing an upper bound of objective difference.

$$\sup_{x \in X} |F(x, \xi_1) - F(x, \xi_2)|$$

Scenario reduction

Implementation error

Definition

Letting $\{\tilde{\xi_1}, ..., \tilde{\xi_K}\}$ be an approximation set of $\{\xi_1, ..., \xi_N\}$, we define the implementation error is:

$$\frac{1}{N}\sum_{i=1}^{N} F(\tilde{x}^*, \xi_i) - \frac{1}{N}\sum_{i=1}^{N} F(x^*, \xi_i) \geq 0 \tag{3}$$

where x^* is the optimal solution of $\{min_x : \frac{1}{N}\sum_{i=1}^{N} F(x, \xi_i)\}$, and $\tilde{x}^*$ is the optimal solution of $\{min_x : \frac{1}{K}\sum_{i=1}^{K} F(x, \tilde{\xi_i})\}$.

Scenario reduction

Implementation error

Upper bound

The algorithm aims to minimize its upper bound:

$$\sum_{k=1}^{K} \sup_{x \in \tilde{X}} \left| F(x, \tilde{\xi}_k) - \frac{1}{|C_k|} \sum_{i \in C_k} F(x, \xi_i) \right| \tag{4}$$

where C_k donates a cluster of scenarios, $\tilde{X}$ donates a hand-made feasible set .

Problems:

- The **the values in a cluster may be very different** but their mean is close to one of them.
- Theoretically, $\tilde{X}$ should includes $x^*, \tilde{x}^*$, but this cannot be guaranteed in practice.

Scenario reduction

Optimization form

Letting $V_{ij} = F(x_i^*, \xi_j)$, where $x_i^* \in argminF(x, \xi_i)$, the clustering problem (4) is equivalent to:

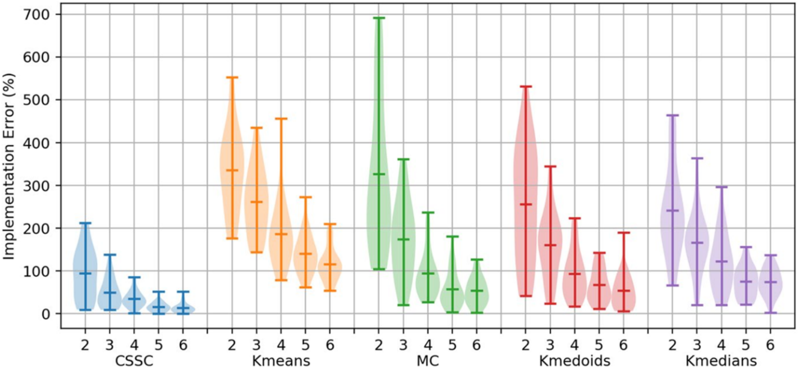
$$
\begin{aligned}
\min_{t_i} \quad & \frac{1}{N}\sum_{i=1}^{N} t_i, \\
\text{s.t.} \quad & t_j \geq \sum_{i=1}^{N} x_{ij} V_{j,i} - \sum_{i=1}^{N} x_{ij} V_{j,j}, \\
& t_j \geq \sum_{i=1}^{N} x_{ij} V_{j,j} - \sum_{i=1}^{N} x_{ij} V_{j,i}, \\
& x_{ij} \leq u_j, \ x_{jj} = u_j, \\
& \sum_{j=1}^{N} x_{ij} = 1, \sum_{j=1}^{N} u_j = K, \\
& all \ for \ i = 1, \dots, N, \ j = 1, \dots, N
\end{aligned}
\tag{5}
$$

where x_{ij} and u_j are binary variables. In practice, there are an approximations:

- $x_i^* \in argminF(x, \xi_i)$ is replaced by its continuous relaxation (because $\mathcal{X}$ is hand-made).

Scenario reduction

Experiments



Since the baselines for comparison are naive, this results are not convincing enough.

Approximation

Using network to approximate complicated terms

- Terms in Objective function
- Terms in traditional algorithm

Approximation

Neur2SP: Neural Two-Stage Stochastic Programming [DPKB22]

Abstract

1) Training network to approximate the second-stage objective value
2) Embedding the network into a MILP
3) Solving the MILP with solver (Gurobi)

Approximation

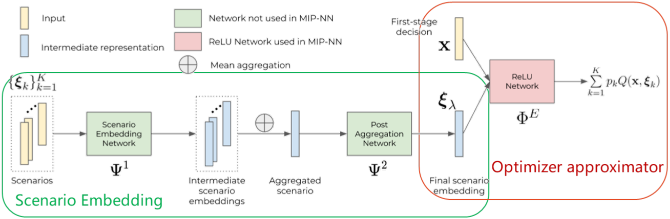
Objective value approximation

$$\mathbb{E}_\xi[Q(x,\xi)] \approx \sum_{k=1}^{K} p_k[Q(x,\xi_k)]$$

- NN-P: for single Q, and then sample scenarios to further approximate.
- NN-E: for whole expectation, sampling during approximation.

Approximation

Model structure



This is the model for the **NN-E** while the NN-P only contains the optimizer approximator, with scenarios inputted directly.
The approximation works for any problem form, i.e. no need to be MILP.

Approximation

Scenario Embedding network

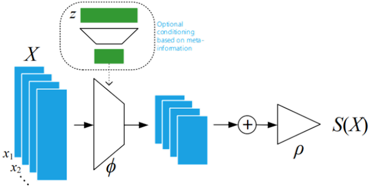


Figure 5: Architecture of DeepSets: Invariant

Deepset

The network is designed to make the input **permutation-invariant**. The set of inputs go through a network parallel independently, and then be merged by mean-aggregation.

Note that this is actually more aggressive scenario reduction and also supervised by the objective value prediction task.

Approximation

Training

- Simple regression
- Data generation:
    - Problems: Capacitated Facility Location Problem (CFLP), Investment Problem (INVP), Stochastic Server Location Problem (SSLP), Pooling Problem (PP)
    1. Random feasible first-stage solution
    2. NN-E: sample K scenarios
       NN-P: sample one scenario
    3. Labeled by Solver

Approximation

After approximation

$$\begin{aligned} \min_{x} \quad & c^\top x + \Phi(x, \xi) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned}$$

- Solving the MINLP with the information of Φ
- network embedding

Approximation

ReLU Network embedding — converting a unit in network into a MIP

For a unit in ReLU network:

$$y = (w^\top x + b)^+,$$

we can rewrite it as:

$$\begin{aligned} w^\top x + b &= \hat{y} - \check{y}, \\ \text{s.t. } \hat{y}, y &\geq 0,\ z \in \{0,1\}, \\ z = 1 &\Longrightarrow \hat{y} \leqslant 0, \\ z = 0 &\Longrightarrow \check{y} \leqslant 0, \end{aligned} \tag{6}$$

The variables are three times as many as the units in the network. This may make the surrogate problem harder than the original one.

Approximation

Experiments

Problem	Data Generation Time		Training Time		Total Time	
	NN-E	NN-P	NN-E	NN-P	NN-E	NN-P
CFLP_10_10	1,823.07	13.59	667.28	127.12	2,490.35	140.71
CFLP_25_25	4,148.83	112.83	2,205.23	840.07	6,354.06	952.90
CFLP_50_50	7,697.91	135.57	463.71	128.11	8,161.62	263.68
SSLP_10_50	942.10	22.95	708.86	116.17	1,650.96	139.13
SSLP_15_45	929.27	16.35	1,377.21	229.42	2,306.48	245.77
SSLP_5_25	860.74	13.18	734.02	147.44	1,594.75	160.62
INVP_B_E	8,951.27	4.17	344.87	1,000.14	9,296.13	1,004.31
INVP_B_H	9,207.90	4.22	1,214.54	607.49	10,422.43	611.71
INVP_I_E	8,759.83	4.34	2,115.25	680.93	10,875.08	685.27
INVP_I_H	8,944.65	3.32	393.82	174.26	9,338.47	177.58
PP	1,202.11	14.86	576.08	367.25	1,778.19	382.11

Table 2: Computing times (in seconds) for data generation and training. Data was generated in parallel with 43 processes.

5000 samples for each instance and NN-E contains multiple scenarios.

Experiments

Problem	Obj. Difference (%)		Solving Time			EF time to	
	EF-NN-E	EF-NN-P	NN-E	NN-P	EF	NN-E	NN-P
INVP_B_E_4	9.54	3.01	0.36	0.34	**0.02**	0.02	0.02
INVP_B_E_9	7.54	2.00	0.31	0.53	**0.04**	0.03	0.03
INVP_B_E_36	2.72	4.96	0.30	9.53	**0.08**	0.02	0.02
INVP_B_E_121	1.37	2.42	**0.33**	86.42	1.69	0.06	0.02
INVP_B_E_441	2.80	2.43	**0.37**	4,342.19	117.59	0.78	1.15
INVP_B_E_1681	1.36	-	**0.34**	-	10,800.01	17.41	0.00
INVP_B_E_10000	-1.48	-	**0.36**	-	10,803.98	-	0.00
INVP_B_H_4	8.81	9.50	0.46	0.25	**0.01**	0.01	0.01
INVP_B_H_9	5.04	5.04	0.30	0.57	**0.03**	0.02	0.02
INVP_B_H_36	1.61	1.61	**0.26**	6.79	1.29	0.01	0.01
INVP_B_H_121	1.77	1.77	**0.33**	45.89	34.69	0.01	0.01
INVP_B_H_441	2.13	5.50	**0.28**	1,870.42	217.46	2.21	0.21
INVP_B_H_1681	-0.71	-	**0.36**	-	10,800.01	-	0.00
INVP_B_H_10000	-2.72	-	**0.36**	-	10,800.03	-	0.00
INVP_I_E_4	12.83	0.00	0.38	0.23	**0.01**	0.01	0.01
INVP_I_E_9	7.40	2.64	0.27	0.35	**0.06**	0.01	0.02
INVP_I_E_36	5.48	5.17	0.27	1.39	**0.04**	0.01	0.01
INVP_I_E_121	5.30	4.49	**0.29**	49.51	1.65	0.02	0.03
INVP_I_E_441	3.00	0.68	**0.26**	2,049.93	46.92	0.08	0.10
INVP_I_E_1681	1.31	3.08	**0.26**	10,834.53	10,800.00	0.41	0.41
INVP_I_E_10000	-1.35	-	**0.30**	-	10,800.10	-	0.00
INVP_I_H_4	13.78	12.16	0.35	0.21	**0.02**	0.01	0.01
INVP_I_H_9	9.12	0.81	0.37	0.31	**0.03**	0.01	0.02
INVP_I_H_36	4.97	3.44	**0.36**	1.99	1.27	0.03	0.03
INVP_I_H_121	4.01	4.99	**0.32**	23.10	7.43	0.07	0.07
INVP_I_H_441	3.15	3.15	**0.32**	1,231.48	10,800.00	0.33	0.33
INVP_I_H_1681	-0.34	0.11	**0.33**	10,816.89	10,800.03	-	252.70
INVP_I_H_10000	-1.60	-	**0.38**	-	10,802.10	-	0.00

Table 5: INVP results: each row represents a single instances. Columns are as in Table 3.

The NN-P underperforms NN-E in terms of both time and objective value.

It could be better to let network do scenario processing.

Approximation

Fast Continuous and Integer L-shaped Heuristics Through Supervised Learning [LFGL22]

Background

- Problem

$$
\begin{aligned}
\min_{x,z,\theta} \quad & cx + dz + \theta \\
\text{s.t.} \quad & Ax + Cz \leq b, \\
& Q(x) - \theta \leq 0, \\
& x \in \{0,1\}^n, \\
& z \geq 0, \quad z \in \mathcal{Z}
\end{aligned}
$$

where $Q(x) = \mathbb{E}_\xi[min_y\{q_\xi y : W_\xi y \geq h_\xi - T_\xi x,\ y \in \mathcal{Y}\}]$.
The L-shaped method aims to get a (x^*, θ^*) by adding cuts to the original problem.

Approximation

Background

- Integer L-shape optimal cut [LL93]

$$(Q(x^*) - L)\Big(\sum_{i \in S(x^*)} x_i - \sum_{i \notin S(x^*)} x_i - |S(x^*)|\Big) + Q(x^*) \leq \theta,$$

where $S(x^*) = \{i : x_i^* = 1\}$.

Although this cut is quite naive, it is the only cut used in the experiment of this paper.

Approximation

Background

- Continuous L-shaped optimal mono-cut [BL11]

$$\mathbb{E}_\xi[\phi_\xi h_\xi] - \mathbb{E}_\xi[\phi_\xi T_\xi]x - \mathbb{E}_\xi[\mathbf{1'}\psi_\xi] \leq \theta$$
$$(\mathbb{E}_\xi[\phi_\xi(h_\xi - T_\xi x) - \mathbf{1'}\psi_\xi] \leq \theta)$$

where ϕ_ξ, ψ_ξ are optimal solutions of dual relaxed sub-problem :
$\{\max_{\phi_\xi,\phi_\xi} \{\phi_\xi(h_\xi - T_\xi x^*) - \mathbf{1'}\psi_\xi : \phi_\xi W_\xi - \psi_\xi \leq q_\xi\}$

This cut is for the second-stage variables y is binary.

Relaxed sub-problem:

$$\begin{aligned} \min_{y} \quad & q_\xi^\top y, \\ \text{s.t.} \quad & W_\xi y \geq h_\xi - T_\xi x^*. \end{aligned} \tag{7}$$

Approximation

Fast Continuous and Integer L-shaped Heuristics Through Supervised Learning [LFGL22]

Abstract

Use network to approximate terms in the cuts:

$$Q(x^*),\ \mathbb{E}_\xi[\phi_\xi h_\xi],\ \mathbb{E}_\xi[\phi_\xi T_\xi]x,\ \mathbb{E}_\xi[\mathbf{1'}\psi_\xi]$$

where the network is fully-connected, with **problem statements as input (including static parameters and random parameters)** and predicted solutions as output.

Approximation

Network setting and results

Data: (x,ξ ,Q(x,ξ))

Problem family	IP/LP	Input length	# Hid. layers	units/ hid. layer	Output length	Abs. rel. error [%]
SSLPF(10,50,2000)	IP	20	10	800	1	0.87
SSLPF-indx(10,50,2000)	IP	20	10	800	1	$\boxed{5.31}$
SSLPF(15,45,15)	IP	30	10	800	1	0.23
SSLPF(15,45,150)	IP	30	10	800	1	0.12
SSLPF(15,80,15)	IP	30	10	800	1	0.40
SMKPF(29)	IP	5	10	800	1	0.071
SMKPF(29)	LP	5	15	1000	7	$\boxed{6.64}$
SMKPF(30)	IP	5	10	800	1	0.072
SMKPF(30)	LP	5	15	1000	7	$\boxed{7.41}$

IP, LP: output is solution of integral or relaxed 2nd stage problem.
Abs. rel. error: average absolute relative prediction error made on ML test set.
The test set is same for SSLPF-indx(10,50,2000) and SSLPF(10,50,2000).

SSLPF: (severs, clients, scenarios), input: (x, ξ), x: whether a server is open, ξ: the server capacities

SMKPF [AAD16]: $x \in \{0,1\}^{100}$, technology matrix $T_{5\times 100}$, input: $Tx + t_{100}$

Approximation

Algorithms

Algorithm 1 Benders decomposition: Main

1: **procedure** MAIN($\boldsymbol{isAlt}, \boldsymbol{\mu}, \boldsymbol{\nu}$)
2: Compute or retrieve the lower bound L for the objective value of (P).
3: Initialize a branch-and-cut process with a global node tree for (M). This creates the repository of leaf nodes, say R. The latter initially contains only the root node.
4: $UB \leftarrow \infty$ ▷ First-stage upper bound
5: $(x^{**}, z^{**}) \leftarrow \varnothing$ ▷ First-stage incumbent solution
6: **if** $R = \emptyset$ **then**
7: **go to** 22
8: **else**
9: Select a node from R.
10: **end if**
11: Compute the current optimal solution (x^*, θ^*) to (M) for the node at hand.
12: **if** $(cx^* + dz^* + \theta^*) \geq UB$ **then**
13: Discard the node from R.
14: **go to** 6
15: **end if**
16: **if** (x^*, z^*) is not integral **then**
17: Partition the domain of (x, z) in (M) or add MIP-based cuts. Accordingly add newly defined nodes to R or update existing nodes in R.
18: **go to** 6
19: **end if**
20: HEURISTICCALLBACK($\boldsymbol{isAlt}, \boldsymbol{\mu}, \boldsymbol{\nu}$)
21: **go to** 6
22: Retrieve the final first-stage incumbent solution (x^{**}, z^{**}). Compute the final overall value $cx^{**} + dz^{**} + Q(x^{**})$.
23: **end procedure**

- The main algorithm follows branch-and-bound paradigm.
- In summary, the step before 20 are to find a feasible (integer constrains) solution under UB.

Approximation

Algorithms

Algorithm 2 Benders decomposition: Heuristic callback

1: **procedure** HEURISTICCALLBACK($\boldsymbol{isAlt}, \boldsymbol{\mu}, \boldsymbol{\nu}$)
2:   **if** !$\boldsymbol{isAlt}$ **then**
3:     **go to** 10
4:   **end if**
5:   Compute predictions ▷ Alternating cut strategy
$\widetilde{Q}^{ML}(x^*)$, $\{\mathbb{E}_\xi[\phi_\xi h_\xi]\}^{ML}$, $\{\mathbb{E}_\xi[\phi_\xi T_\xi]\}^{ML}$, $\{\mathbb{E}_\xi[\mathbf{1}'\psi_\xi]\}^{ML}$
or
$\widetilde{Q}^{ML}(x^*)$, $\{\mathbb{E}_\xi[\phi_\xi]\}^{ML}$, $\{\mathbb{E}_\xi[\mathbf{1}'\psi_\xi]\}^{ML}$.
6:   **if** $\nu\widetilde{Q}^{ML}(x^*) > \theta^*$ **then**
7:     Add a heuristic continuous L-shaped mono-cut (14) or (15).
8:     **return**
9:   **end if**
10:   Compute prediction $Q^{ML}(x^*)$. ▷ Integer L-shaped method
11:   **if** $\mu Q^{ML}(x^*) \leq \theta^*$ **then**
12:     **if** $cx^* + dz^* + \theta^* < UB$ **then**
13:       $UB \leftarrow cx^* + dz^* + \theta^*$ ▷ Update upper bound
14:       $(x^{**}, z^{**}) \leftarrow (x^*, z^*)$ ▷ Update incumbent solution
15:     **end if**
16:   **else**
17:     Add a heuristic integer L-shaped cut (12).
18:   **end if**
19: **end procedure**

- The μ, ν are set in $[0, 1]$ and close to 1. Since it is a minimizing problem, the predicted $\tilde{Q}^{ML}(x^*), Q^{ML}(x^*)$ overestimate the exact value.
- Once set the **isAlt** false, we do not need to predict the solution

Approximation

Experiments

Problem family	Alt-L Quantiles 0.05	Alt-L Quantiles 0.5	Alt-L Quantiles 0.95	Alt-L Avg	ML-L-Shaped Quantiles 0.05	ML-L-Shaped Quantiles 0.5	ML-L-Shaped Quantiles 0.95	ML-L-Shaped Avg	Optimality gap Quantiles 0.05	Optimality gap Quantiles 0.5	Optimality gap Quantiles 0.95	Optimality gap Avg
SSLPF(10,50,2000)	-353.85	-347.14	-333.17	-345.60 (0.70)	-353.85	-347.14	-333.17	-345.58 (0.70)	0.000%	0.000%	0.036%	0.006% (0.003%)
SSLPF-indx(10,50,2000)	-353.85	-347.14	-333.17	-345.60 (0.70)	-348.92	-339.00	-315.25	-336.57 (1.06)	0.000%	2.173%	7.850%	2.609% (0.242%)
SSLPF(15,45,15)	-313.20	-308.74	-296.62	-307.15 (0.59)	-313.20	-308.67	-296.48	-306.95 (0.60)	0.000%	0.000%	0.608%	0.064% (0.019%)
SSLPF(15,45,150)	-314.16	-306.42	-294.74	-305.89 (0.66)	-314.15	-306.27	-281.51	-300.01 (3.61)	0.000%	0.000%	1.021%	1.943% (1.150%)
SSLPF(15,80,15)	-632.21	-614.67	-584.42	-613.43 (1.34)	-630.52	-614.67	-584.42	-612.97 (1.34)	0.000%	0.000%	0.538%	0.075% (0.018%)
SMKPF(29)	7736.92	8176.1	8567.36	8155.42 (27.60)	7736.92	8178.5	8570.25	8156.18 (25.58)	0.000%	0.000%	0.050%	0.009% (0.002%)
SMKPF(30)	8288.24	8790.53	9160.83	8754.33 (28.18)	8228.24	8790.53	9164.16	8754.73 (28.19)	0.000%	0.000%	0.027%	0.005% (0.001%)

Standard error of estimate is reported between parentheses.

Table 3: First-stage values and optimality gaps

This paper only conduct the comparison for the false-isAlt version, which means no solution prediction.

Summary

Scenario reduction

- vector difference
- objective value difference

Problems: ignoring constraints, ill mapping: $\xi \to (x, y)$
Possible direction: Measurement for similarity of optimization problems

Approximation

- Terms in Objective function
- Terms in traditional algorithm

Problems: embedding problems (ReLU, size), unreliable solution prediction
Possible direction: Solving MINLP without network embedding

References I

[AAD16] Gustavo Angulo, Shabbir Ahmed, and Santanu S Dey, *Improving the integer l-shaped method*, INFORMS Journal on Computing **28** (2016), no. 3, 483–499.

[BL11] John R Birge and Francois Louveaux, *Introduction to stochastic programming*, Springer Science & Business Media, 2011.

[DPKB22] Justin Dumouchelle, Rahul Patel, Elias B Khalil, and Merve Bodur, *Neur2sp: Neural two-stage stochastic programming*, arXiv preprint arXiv:2205.12006 (2022).

[KOR23] Julien Keutchayan, Janosch Ortmann, and Walter Rei, *Problem-driven scenario clustering in stochastic optimization*, Computational Management Science **20** (2023), no. 1, 13.

References II

[LFGL22] Eric Larsen, Emma Frejinger, Bernard Gendron, and Andrea Lodi, *Fast continuous and integer l-shaped heuristics through supervised learning*, arXiv preprint arXiv:2205.00897 (2022).

[LL93] Gilbert Laporte and François V Louveaux, *The integer l-shaped method for stochastic integer programs with complete recourse*, Operations research letters **13** (1993), no. 3, 133–142.

[WSCZ22] Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang, *Learning scenario representation for solving two-stage stochastic integer programs*, International Conference on Learning Representations, 2022.