Learning Disentangled Representations of Stochastic Programs

Benjamin Hudson
Mila & DIRO, Université de Montréal
Montreal, Canada
ben.hudson@mila.quebec

Abstract

Two-stage stochastic programming is a prevalent framework for optimal decision-making under uncertainty. This paper investigates learning a model of the second stage in a two-stage program from historical decision data. We propose a novel formulation of this inverse optimization problem as a causal representation learning task, enabling the recovery of distributions over the problem's latent constraint levels and point estimates of its constraint matrices. We compare three VAE-based models, two of which ensure disentanglement, and demonstrate the effectiveness of our approach in learning disentangled representations of the latent parameters. The results highlight the potential of causal representation learning in improving the interpretability and computational efficiency of learned linear programming models.

1 Introduction

Two-stage stochastic programming is a powerful framework used for making optimal decisions under uncertainty, prevalent in operations research and widely applied in practice. In this approach, decision-making is split into two stages: a "here-and-now" decision is made first, before uncertain quantities are unrealized, followed by a "recourse" decision that is determined once the uncertainty is realized. The objective is to minimize the total cost, which is incurred by decisions made at both stages. Two-stage stochastic programming finds applications in supply chain management, financial planning, and energy distribution, where decisions must be robust across a range of possible future scenarios. A key challenge in two-stage stochastic programming is formulating models that faithfully represent the real-world problem and the distributions of random variables, yet remain computationally tractable.

Goals and Contributions In this paper, we investigate a data-driven approach to this challenge. Using a dataset of historical decisions from a decision-making process that can be modelled as a two-stage stochastic programming problem, we recover a model of the second-stage problem and distributions over its latent parameters. Our contributions are:

- A formulation of this inverse optimization problem as a causal representation learning task.
- An method to recover the latent parameters of the second-stage problem, including distributions over the latent constraint levels $h(\xi)$ and point estimates of the constraint matrices T and W, from a dataset of historical decisions.
- Empirical results of applying this method to the task of recovering the latent constraint levels of a synthetic two-stage stochastic programming problem. We compare two VAE-based models that guarantee disentanglement and one that does not.

Preprint. Under review.

Paper Layout The rest of the paper is laid out as follows. The next section describes related work. The third section provides background on linear programming and two-stage stochastic programming, including relevant notation. The fourth section describes how we formulate the problem as a causal representation learning problem, and our approach to recovering the latent parameters of the second-stage problem. The fifth section details our empirical studies, the models we applied and their key assumptions, and presents and discusses our experimental results. Finally, the sixth section concludes our paper and outlines future work.

2 Related Work

In operations research, *inverse optimization* describes the task of fitting mathematical optimization model to decision data by learning the objectives and constraints of the underlying agent behaviour. Chan et al. [3] divides inverse optimization methods into "classical" and "data-driven" approaches, where the former assume perfect model-data fit and the latter does not. Classical problems, which dominate the early literature on inverse optimization, often introduce new formulations of the inverse problem itself. On the other hand, data-driven models draw upon the methods employed in classical approaches but introduce an additional layer of complexity arising from noisy data.

Our two-stage stochastic problem is noisy by definition, and therefore fits into the data-driven category. Also in this category, Aswani et al. [1] consider a setting where a dataset of optimal solutions to a convex optimization problem is corrupted by noise. They provide a method for solving the inverse optimization problem in this setting and prove its statistical consistency. Tan et al. [11] study inverse optimization in a contextual optimization setting; a setting where the latent problem variables change depending on some context variables. They introduce an inverse optimization method for learning linear programs in this setting. Besbes et al. [2] also study a contextual optimization setting, but consider more general non-linear decision-making problems. They aim to find a decision-making policy rather than recover the objectives and constraints of the underlying problem. Dong and Zeng [4] consider general multi-objective non-linear decision-making problems. They formulate the inverse optimization problem as an unsupervised learning task and provide two algorithms to solve it.

The main difference between our paper and the works above is that we aim to recover *distributions* over latent problem variables, whereas other inverse optimization methods recover point estimates. Distributional information on problem parameters is essential for safe and effective decision-making in the real world. We focus on two-stage problems, but our method can also be applied to a contextual setting by replacing the first-stage decisions with contexts as "perturbations" to the second-stage stochastic problem.

3 Preliminaries

We will focus on two-stage stochastic programming problems in which both stages are represented by linear programs. We will briefly recall linear programming, necessary notation, then introduce two-stage stochastic programming.

3.1 Linear Programming

Linear programming is a widely-used method for optimizing a linear objective function, subject to linear constraints. We will focus on minimization in this section, but $\min$ and $\max$ are interchangable. A linear program typically consists of four parts:

Decision Variables The variables that are optimized over. For example, x_1 and x_2.

An Objective Function The linear function to minimize. For example, $f(x_1, x_2) = c_1x_1 + c_2x_2$.

Problem Constraints Linear inequality constraints on the objective function. For example,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 &\leq b_2 \\ a_{31}x_1 + a_{32}x_2 &\leq b_3. \end{aligned}$$

Non-negativity Constraints All decision variables are subject to non-negativity constraints. In our example, $x_1 \geq 0$ and $x_2 \geq 0$.

Any problem constraint that is represented by an inequality can be converted to an equality constraint by introducing an additional decision variable that represents the offset (or "slack") to the constraint level. For example,

$$a_{11}x_1 + a_{12}x_2 \leq b_1 \Leftrightarrow a_{11}x_1 + a_{12}x_2 + x_3 = b_1.$$

The additional slack variable does not contribute to the objective function, but is subject to non-negativity constraints. Converting all inequality constraints to equality constraints gives the *augmented form* of a linear program (as opposed to the *standard form*, which uses inequality constraints). We will use the augmented form in the remainder of the paper, which can be expressed compactly using matrices as

$$\begin{aligned} \min_x \; & f(x) := c^T x \\ \text{s.t. } & Ax = b \\ & x \geq 0. \end{aligned} \tag{1}$$

where x is a column vector of decision variables including slack variables, c is a column vector of costs, A is a matrix representing how each decision variable contributes to each constraint, and b is a column vector of constraint levels.

3.2 Two-stage Stochastic Programming

As its name implies, two-stage stochastic programming splits decision-making into two stages. We focus on problems where both stages are represented by linear programs. The first stage represents a decision-making problem that must be solved (i.e. a decision must be taken) before uncertain quantities are realized. The second stage represents a problem that can be solved after uncertain quantities are realized. The second-stage decision can mitigate or adjust the decision made at the first stage, which is why it is sometimes called a "recourse" decision. The objective is to minimize the total cost incurred by both decisions.

The first-stage problem takes the uncertainty in the second-stage problem into account in order to minimize the cost incurred by the eventual recourse decision. The first-stage decision-making problem is formulated as

$$\begin{aligned} \min_x \; & c^T x + \mathbb{E}_\xi[Q(x,\xi)] \\ \text{s.t. } & Ax = b \\ & x \geq 0, \end{aligned} \tag{2}$$

where $\mathbb{E}_\xi[Q(x,\xi)]$ represents the expected cost of the second-stage decision. The cost depends on the first-stage decision x and some realization of random variables ξ. The second-stage problem formulation is similar to the first. It is

$$\begin{aligned} \min_y \; & q(\xi)^T y \\ \text{s.t. } & T(\xi)x + W(\xi)y = h(\xi) \\ & y \geq 0, \end{aligned} \tag{3}$$

where $y, q(\xi), W(\xi)$ and $h(\xi)$ are equivalent to x, c, A and b in the first-stage problem, but stochastic. $T(\xi)$ represents how the first-stage decision impacts the second-stage problem. Skip to Section 5.1 for a specific, tangible example of a two-stage stochastic programming problem.

4 Learning Causal Representations of Stochastic Programs

We aim to recover a model of the second-stage problem and the distributions of its latent parameters $q(\xi), h(\xi), W(\xi)$, and $T(\xi)$ from a dataset of first- and second-stage decisions $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$. In this work, we focus on learning the constraint levels $h(\xi)$ only, as these often represent interesting real-world quantities.

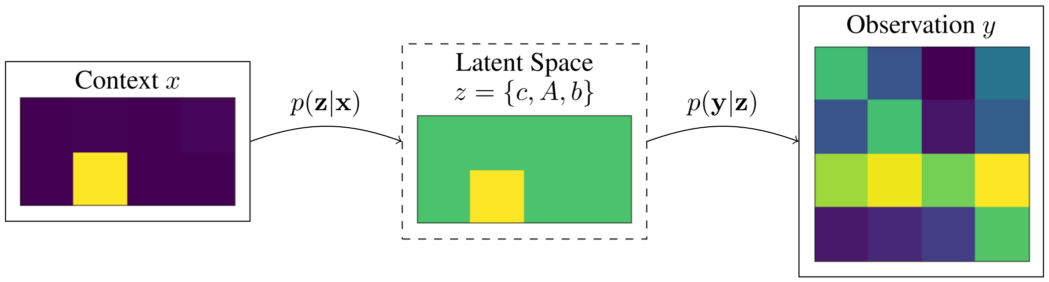

Figure 1: Schematic of our approach to learning representations of stochastic programs. For a given context x, the latent variables z are drawn from the prior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. Then, the optimal decisions y are generated from the distribution $p_\theta(\mathbf{y}|\mathbf{z})$.

4.1 Representation Learning Formulation

To frame our task as a representation learning problem, we essentially apply the reparameterization trick to the second-stage program in reverse. We take the perspective that the second-stage program is a deterministic, differentiable function mapping a vector of first-stage decisions x and exogenous noise ξ to a vector of optimal second-stage decisions y. The program's latent variables $z = \{q, T, W, h\}$ can be calculated as deterministic functions of the noise and the parameters of their distributions.

Now, applying the trick in reverse, we change our perspective to thinking about the second-stage problem in terms of probability distributions $p_\theta(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z})$. The data-generating process, shown in Figure 1, is as follows: for a given first-stage decision x, the latent variables z are drawn from the prior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. Then, the second-stage decisions y are generated from the distribution $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z})$. Following Khemakhem et al. [5], we make the assumption that $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{y}|\mathbf{z})$. We must learn a decoder to approximate $p_\theta(\mathbf{z}|\mathbf{x})$ and an encoder or "mixing function" to approximate $p_\theta(\mathbf{y}|\mathbf{z})$ from data.

For the learned representation to be useful, it must have the form in eq. (3), and therefore we must recover the various latent variables explicitly. In this work, we focus on learning distributions over the latent constraint levels only, i.e. $z = \{h\}$. We do not attempt to recover the objective function coefficients q. Instead, we force q to be deterministic and make it part of the mixing function that renders y from z. It is possible to recover point estimates for the transition matrices T and W. If we assume they are binary, as is the case in many two-stage stochastic programs, we can recover them from $p_\theta(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{y}|\mathbf{z})$ respectively. We recover T by testing independence between each element of x and h: $x_i \not\perp\!\!\!\perp h_j$ indicates $T_{ij} = 1$, otherwise it is 0. Similarly, $y_i \not\perp\!\!\!\perp h_j$ indicates $W_{ij} = 1$, otherwise it is 0. In conclusion, we are able to recover the distribution of the latent constraint levels $h(\xi)$, as well as point estimates of the matrices T and W, provided they are binary.

4.2 Estimation by VAE

Our formulation clearly suggests a VAE-based approach. We experiment with three VAE-based models in Section 5, two of which guarantee disentanglement up to permutation and scaling, and one which does not.

4.3 Motivation for a Causal Perspective

We have two incentives to take a causal perspective to this problem. Firstly, it is very important that the learned representation be as sparse as possible for downstream tasks. Two-stage stochastic programs are solved by sampling many "scenarios" from the distributions of the second-stage problem and combining them into a single, super-program called the *Deterministic Equivalent Program* (DEP). A DEP with more scenarios yields a higher-fidelity decision, but is also more computationally complex. Sparsity, especially in the constraint matrices T and W, reduces the computational complexity of the problem. The ground truth representations are often sparse, and therefore we are motivated to learn un-rotated representations. Linear programs are agnostic to permutation and scaling of their parameters, which makes causal representation learning a great fit for this task.

Our second incentive for taking a causal perspective to this problem comes from interpretability. The latent variables q and h often represent interesting real-world quantities. For example, in a transportation problem q could represent empirical travel times between locations. Thus, learning disentangled representations of these variables is of extra value to the decision-maker.

5 Empirical Studies

5.1 Taxi Allocation and Dispatching Problem

We study a daily allocation and dispatching problem faced by a hypothetical taxi company. The company's service area is divided into many zones. Before each day of service, the company must decide how to allocate their budget: they can schedule drivers who will be positioned to serve demand in a certain zone, and they can create promotions to induce more demand in some zones. On each day of service, once the drivers are scheduled and the promotions are published, the company must decide how to dispatch drivers from the zones where they were positioned to serve realized passenger demand. The company's objective is to maximize the demand they serve while minimizing the costs they incur from scheduling, promotions, and dispatching. This problem can be modelled as a two-stage stochastic programming problem in which the first stage represents the budget allocation problem and the stochastic second stage represents the dispatching problem. Our goal is to use a dataset of historical allocation and dispatching decision data (x and y) to recover distributions over the constraint levels in the second-stage problem ($h(\xi)$), representing the number of drivers available for dispatching and the passenger demand in each of the zones in the service area. Appendix A has a complete mathematical formulation of the two-stage stochastic programming problem.

Synthetic Environment We use a synthetic environment with four zones. There are two constraint levels per zone: the number of drivers available for dispatching (or simply "capacity") and passenger demand. These are normally distributed around around their target levels, which is set by the decision-maker at the first stage, by scheduling or targeted promotions. The uncertainty in the capacity level can be attributed to drivers being hailed by passers-by, vehicle breakdowns, or being otherwise unavailable. The travel times between zones are deterministic.

Sampling Strategy A sample in the dataset is generated from the environment as follows. First, we create a random solution to the first-stage budget allocation problem. We schedule a base level of capacity in all zones that we expect is sufficient to meet passenger demand, and we leave demand unperturbed. Then, we use the remaining budget to either increase or suppress the mean capacity or demand level in up to three random zones by some fixed amount (this is the perturbation strategy from Lopez et al. [8]). These first-stage decisions can be thought of as random interventions that perturb the latent parameters (i.e. the capacity and demand levels) of the second-stage problem. We draw samples from the resulting capacity and demand distributions, and solve the second-stage problem optimally. We record the first- and second-stage decisions (including the slack decision variables representing unused capacity and unserved demand in each zone) in the dataset. Figure 2 shows two samples from the environment. Our dataset contains 1000 random first-stage decisions, each sampled 1000 times, for a total of 1M samples.

5.2 Models

We compare three VAE-based models for this task, two of which guarantee disentanglement up to permutation and scaling (iVAE from Khemakhem et al. [5] and sVAE from Lachapelle et al. [6]) and one which does not (cVAE from Sohn et al. [10]). All three offer a way to condition the distribution of the latent variables on some auxiliary information, which is the first-stage decision in our experiments. We use the "action-sparsity" version of the sVAE because our problem does not have dependencies across time.

The iVAE and sVAE models make many assumptions about the data-generating process to prove they learn disentangled representations. We summarize what we believe to be the most important assumptions and discuss them in the context of our taxi environment.

Assumptions on the Mixing Function Firstly, both models assume that the mixing function is invertible. This is the case in our environment when we include the slack variables in the observations

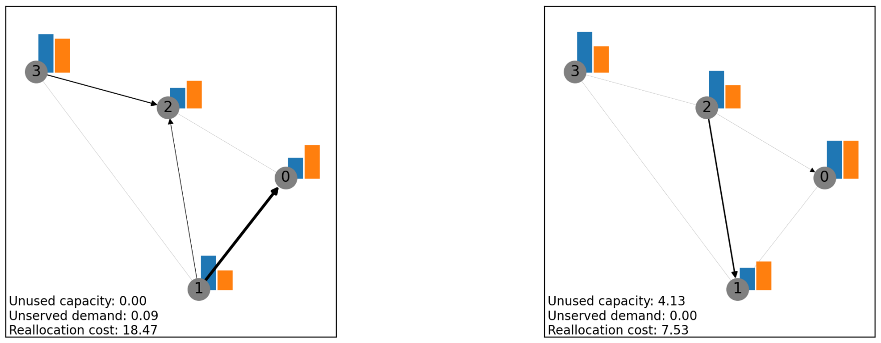

Figure 2: Two random samples from our synthetic taxi allocation and dispatching environment. The blue and orange bars adjacent to each node represent the number of taxis scheduled and the amount of demand in that zone, respectively. These are latent constraint levels h. They are perturbed by random first-stage decisions x, which are observed. The thickness and direction of the arrows between nodes represent the number of taxis dispatched in order to minimize the travel time and unserved demand. These represent the observed second-stage decisions y.

of the second-stage decisions. Furthermore, iVAE requires that the mixing function is injective, and sVAE requires that it is bijective. In our environment, the mixing function is the second-stage program which is injective. Therefore sVAE's assumption is *not* satisfied.

Distributional Assumptions on the Latents Next, both models require that the latent factors follow a distribution from the exponential family. Furthermore, iVAE requires that latent factors have more than one sufficient statistic, or they have one, non-monotonic sufficient statistic. In the action-sparsity case, sVAE requires that the latent factors have exactly one sufficient statistic. The latent factors in our environment are fixed-variance Gaussians, and thus have one sufficient statistic. The means do not depend on the past, and are therefore non-monotonic (Lachapelle et al. [6], Appendix A.6). Therefore, the distributional assumptions on the latent variables are satisfied.

Graphical Criterion Finally, the sVAE makes one additional assumption, which the authors call the "graphical criterion". In the action-sparsity case, it states that for each latent variable, there must exist some subset of interventions in the dataset such that that latent variable is the only intervened variable in common across the interventions in the subset. Intuitively, this means that the model is able to single out the effect on each latent alone using some subset of interventions. In our dataset, each intervention perturbs up to three latents (target capacity or demand level). Given 1000 random interventions, the likelihood of two of eight latent variables being *only* intervened on together is very low, so it is reasonable to assume this assumption is satisfied.

5.3 Experimental Setup

We used open-source implementations of iVAE[1] and sVAE[2] and our own PyTorch implementation of cVAE, as no open-source one was available. We set the latent dimension to the ground truth value (eight) in all models. We trained all models using their default hyperparameters, with the following exceptions:

- We set the hidden dimension in all MLPs in iVAE to 64.
- We trained sVAE using the constrained optimization strategy described in [7]. We set the maximum number of edges in the action graph G_a to the ground truth value (eight).

We trained all three models on the same dataset using a train-validation-test split of 0.8-0.1-0.1. We trained them on single-GPU nodes of Mila's cluster for up to 3 hours or 500 epochs, whichever was less. We ran five seeds per experiment.

[1] `https://github.com/ilkhem/icebeem/tree/master/models/ivae`

[2] `https://github.com/slachapelle/disentanglement_via_mechanism_sparsity`

Model	R^2	MCC
iVAE ([5])	0.78 ± 0.01	$\mathbf{0.61 \pm 0.05}$
sVAE ([6])	0.70 ± 0.06	0.53 ± 0.10
cVAE ([10])	$\mathbf{0.95 \pm 0.03}$	0.57 ± 0.02

Table 1: Experimental Results

Metrics We measure model performance based on two disentanglement metrics. The R^2 score evaluates how well a linear combination of the learned latent variables predicts the ground truth ones. It is a measure of linear disentanglement. The Mean Correlation Coefficient (MCC) score measures permutation disentanglement: how well the model learns a representation that is disentangled "up to permutation and scaling". Scores closer to 1 are better for both metrics.

5.4 Results & Discussion

Table 1 shows the results of our experiments. Interestingly, cVAE acheives a similar MCC score to the models that guarantee disentanglement up to permutation and scaling. By all measures, it learns a more disentangled representation than sVAE. Furthermore, its performance is also relatively unaffected when the mixing function is uninvertible (see Appendix B.1).

Looking more closely, we find that iVAE and cVAE are identical, with the exception of one term in the training objective. iVAE is trained on the prior-posterior KL divergence alone, and cVAE is trained on this and an additional reconstruction loss term (MSE between the original and reconstructed second-stage decisions). Since iVAE guarantees disentanglement by making assumptions about the data-generating process and not the model itself, we can conclude that cVAE actually learns a disentangled representations as well. Its superior performance can probably be attributed to the additional loss term in the training objective.

6 Conclusion & Future Work

We investigated using a dataset of historical decisions from an unknown two-stage stochastic programming problem to recover a model of the second-stage problem and distributions over its latent parameters. We introduced a formulation of this inverse optimization problem as a causal representation learning task, as well as a method to recover the latent parameters of the second-stage problem, including distributions over the latent constraint levels $h(\xi)$ and point estimates of the constraint matrices T and W. Finally, we evaluated three VAE-based models on this task in a synthetic two-stage stochastic programming environment representing a taxi allocation and dispatching problem.

Extending our problem formulation to also recover the latent cost distributions $q(\xi)$ is very important for our approach to be useful in practice. This represents the most important piece of future work. Additionally, Equation (8) in Appendix A clearly shows that the effect of the latent variables on the second-stage decisions is sparse. Unfortunately, of the methods we surveyed, only Moran et al. [9] leverages sparsity in the decoding function. This method requires for each latent, there at there exist at least two features that depend on that latent alone (the authors call these "anchor features"). This is not not the case in our problem, as every "feature" y_{ij} is depends on a capacity constraint level and a demand constraint level. However, we think there is an opportunity to leverage action-sparsity and decoding-sparsity together, which is the case in our two-stage stochastic programming environment. We leave this to future work.

References

[1] Anil Aswani, Zuo-Jun Max Shen, and Auyon Siddiq. *Inverse Optimization with Noisy Data*. 2017. arXiv: 1507.03266 [math.OC].

[2] Omar Besbes, Yuri Fonseca, and Ilan Lobel. "Contextual Inverse Optimization: Offline and Online Learning". In: *Operations Research* (2023). DOI: 10.1287/opre.2021.0369. eprint: https://doi.org/10.1287/opre.2021.0369. URL: https://doi.org/10.1287/opre.2021.0369.

[3] Timothy C. Y. Chan, Rafid Mahmood, and Ian Yihang Zhu. “Inverse Optimization: Theory and Applications”. In: *Operations Research* (2023). DOI: `10.1287/opre.2022.0382`. eprint: `https://doi.org/10.1287/opre.2022.0382`. URL: `https://doi.org/10.1287/opre.2022.0382`.

[4] Chaosheng Dong and Bo Zeng. “Expert Learning through Generalized Inverse Multiobjective Optimization: Models, Insights, and Algorithms”. In: *Proceedings of the 37th International Conference on Machine Learning* (July 13, 2020). Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2648–2657. URL: `https://proceedings.mlr.press/v119/dong20f.html`.

[5] Ilyes Khemakhem et al. “Variational Autoencoders and Nonlinear ICA: A Unifying Framework”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* (Sept. 26, 2020). Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2207–2217. URL: `https://proceedings.mlr.press/v108/khemakhem20a.html`.

[6] Sebastien Lachapelle et al. “Disentanglement via Mechanism Sparsity Regularization: A New Principle for Nonlinear ICA”. In: *Proceedings of the First Conference on Causal Learning and Reasoning* (Apr. 11, 2022). Vol. 177. Proceedings of Machine Learning Research. PMLR, 2022, pp. 428–484. URL: `https://proceedings.mlr.press/v177/lachapelle22a.html`.

[7] Sébastien Lachapelle et al. *Nonparametric Partial Disentanglement via Mechanism Sparsity: Sparse Actions, Interventions and Sparse Temporal Dependencies*. 2024. arXiv: `2401.04890 [stat.ML]`.

[8] Romain Lopez et al. “Learning Causal Representations of Single Cells via Sparse Mechanism Shift Modeling”. In: *Proceedings of the Second Conference on Causal Learning and Reasoning* (Apr. 11, 2023). Vol. 213. Proceedings of Machine Learning Research. PMLR, 2023, pp. 662–691. URL: `https://proceedings.mlr.press/v213/lopez23a.html`.

[9] Gemma Elyse Moran et al. “Identifiable Deep Generative Models via Sparse Decoding”. In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: `https://openreview.net/forum?id=vd0onGWZbE`.

[10] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. “Learning Structured Output Representation using Deep Conditional Generative Models”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: `https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf`.

[11] Yingcong Tan, Daria Terekhov, and Andrew Delong. “Learning Linear Programs from Optimal Decisions”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19738–19749. URL: `https://proceedings.neurips.cc/paper_files/paper/2020/file/e44e875c12109e4fa3716c05008048b2-Paper.pdf`.

A Taxi Problem Formulation

Here we will provide a complete mathematical formulation of the two-stage stochastic programming problem described in Section 5.1. This is the ground-truth representation that we aim to recover using our method. We will begin with an intuitive formulation and convert it to the augmented form.

The first-stage decision variables represent the amount of budget allocated to scheduling drivers and creating promotions in each zone. There is a single constraint, which states that the total amount allocated must not exceed the available budget. Finally, the objective function includes the cost of allocating drivers to a given location, the cost of creating promotions in that location, and the expected cost of the second-stage problem. Mathematically,

$$\begin{aligned} \min_x \ & \sum_i x_i + \mathbb{E}_\xi[Q(x,\xi)] \\ \text{s.t.} \ & \sum x_i \le b \\ & x \ge 0, \end{aligned} \tag{4}$$

where x is an eight-dimensional column vector containing the amount of budget allocated to scheduling drivers and creating promotions at each of the four zones. We can convert the inequality constraint to an equality constraint by adding a slack variable u to represent the amount of unallocated budget. The augmented form of the first-stage problem is

$$\begin{aligned} \min_{x}\ & \mathbf{1}x + \mathbb{E}_{\xi}[Q(x,\xi)] \\ \text{s.t. } & \mathbf{1}x + u = b \\ & x \geq 0,\ u \geq 0, \end{aligned} \tag{5}$$

where $\mathbf{1}$ is an 8-dimensional vector of ones.

The second-stage problem is formulated as

$$\begin{aligned} \min_{y}\ & \sum_i \sum_j q_{ij} y_{ij} + p_i v_i \\ \text{s.t. } & \sum_j y_{ij} \leq c(x_i,\xi) \quad i = 1,\ldots,4 \\ & \sum_i y_{ij} + v_j = d(x_{j+4},\xi) \quad j = 1,\ldots,4 \\ & y \geq 0 \end{aligned} \tag{6}$$

where y is a 4×4 matrix containing the number of taxis dispatched from zone i to zone j, q is a 4×4 travel time matrix, v is a 4-dimensional column vector containing the amount of unserved demand at each zone, and p is the penalty per unit of unserved demand per zone. The objective of the problem is to maximize the total demand served while minimizing dispatching travel time. The first constraint states that the number of taxis dispatched from a zone may not exceed the available capacity in that zone. Available capacity is determined by a "capacity function" c that accepts the number of drivers allocated in the stage decision x_i and some random noise ξ. Similarly, the second constraint states that the number of taxis dispatched *to* a zone plus the unserved demand must not exceed the total demand in each zone. The total demand is determined by some "demand function" d that accepts the promotions for that zone x_{j+4} and some noise ξ (the offset is required because x is a column vector containing the scheduling decisions and then the promotion decisions). The v variable is necessary to preserve the linearity of the problem.

We can introduce an additional decision variable w to transform the first inequality constraint into an equality constraint.

$$\sum_j y_{ij} \leq c(x_i,\xi) \Leftrightarrow \sum_j y_{ij} + w_i = c(x_i,\xi) \tag{7}$$

To transform this problem into the augmented form, we must first flatten y into a 16-dimensional column vector. Flattening y in row-major order allows us to express the constraints in matrix form,

$$\begin{aligned} \min_{y}\ & q^T y + p^T u \\ \text{s.t. } & \left[\begin{array}{cccc|c} \mathbf{1} & & & & \\ & \mathbf{1} & & & \\ & & \mathbf{1} & & \mathbf{I}_{8,8} \\ & & & \mathbf{1} & \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \end{array}\right] \begin{bmatrix} y \\ v \\ w \end{bmatrix} = \begin{bmatrix} c(x_{1:4},\xi) \\ d(x_{5:8},\xi) \end{bmatrix} \\ & y \geq 0,\ v \geq 0,\ w \geq 0, \end{aligned} \tag{8}$$

where $\mathbf{1}$ represents a 4-dimensional vector of ones and $\mathbf{I}$ represents a 4-dimensional identity matrix. Defining $h(x,\xi) = [c(x_{1:4},\xi)\ d(x_{5:8},\xi)]^T$ gives us something close Equation (3), but without $W(\xi)$ explicitly.

B Additional Experiments

B.1 Unobserved Slack Variables

Removing the slack variables (unused capacity and unserved demand) is more realistic; it is unlikely that the decision-maker could know the amount of unserved demand on a given day of service. However, it also makes the rendering function uninvertible, which both iVAE and sVAE require for disentanglement. We experimented with this setup, and the performance of all three models is shown below. The performance of iVAE and sVAE decreases considerably, but that of cVAE is relatively unaffected.

Model	R^2	MCC
iVAE ([5])	0.67 ± 0.02	0.45 ± 0.03
sVAE ([6])	0.63 ± 0.02	0.41 ± 0.03
cVAE ([10])	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.56 \pm 0.03}$