LEARNING SCENARIO REPRESENTATION FOR SOLVING TWO-STAGE STOCHASTIC INTEGER PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many practical combinatorial optimization problems under uncertainty can be modeled as stochastic integer programs (SIPs), which are extremely challenging to solve due to the high complexity. To solve two-stage SIPs efficiently, we propose a conditional variational autoencoder (CVAE) based method to learn scenario representation for a class of SIP instances. Specifically, we design a graph convolutional network based encoder to embed each scenario with the deterministic part of its instance (i.e. context) into a low-dimensional latent space, from which a decoder reconstructs the scenario from its latent representation conditioned on the context. Such a design effectively captures the dependencies of the scenarios on their corresponding instances. We apply the trained encoder to two tasks in typical SIP solving, i.e. scenario reduction and objective prediction. Experiments on two SIP problems show that the learned latent representation significantly boosts the solving performance to attain high-quality solutions in short computational time, and generalizes fairly well to problems of larger sizes or with more scenarios.

1 INTRODUCTION

Stochastic integer programs (SIPs) are a class of combinatorial optimization problems (COPs) with uncertain parameters. They focus on finding a best decision for a practical task to optimize both the deterministic cost and the expectation of stochastic costs on a set of potential *scenarios*. Since the uncertain elements commonly exist in reality, SIPs are widely applied and studied in many fields such as transportation, inventory management, energy optimization, and so on (Jammeli et al., 2019; Dillon et al., 2017; Bruno et al., 2016; Wallace & Ziemba, 2005). However, optimally solving SIP is intractable especially when many scenarios are involved, due to the fact that: 1) most integer programs themselves are already NP-hard; 2) considerable (even continuous) scenarios significantly increase the computational complexity (Klein, 2021). In reality, solving moderate-sized problems may require prohibitively long time. Thus efficient and high-quality approximate solutions for SIPs are often practically pursued.

The advances in deep (reinforcement) learning has aroused extensive studies on solving optimization problems with neural networks. The related works have shown promising results in tackling some classic COPs, e.g., vehicle routing, job shop scheduling, combinatorial auction, etc. (Kool et al., 2019; Zhang et al., 2020; Lee et al., 2020; Kwon et al., 2020; Gasse et al., 2019; Chen & Tian, 2019). However, most of the existing deep learning based methods only focus on deterministic problems, which could hinder their practical applications in the uncertain situation. On the other hand, a few early attempts introduce deep learning to solve SIPs (Larsen et al., 2018; Abbasi et al., 2020; Joe & Lau, 2020; Bengio et al., 2020; Nair et al., 2018). However, they are generally designed for only specific problems and also lack the ability of generalizing across different problem sizes.

In this paper, we propose to learn the deep latent representation of *scenarios* in general SIPs, which can be applied to various downstream tasks in solving SIPs, e.g., *scenario reduction* and *objective prediction*. Based on the basic two-stage form of SIPs, we leverage conditional variational autoencoder (CVAE) to learn latent and continuous representations of stochastic scenarios while considering the deterministic part of a SIP instance, which we term *context*. Specifically, we employ a graph convolutional network (GCN) to learn embeddings of each deterministic integer program, which models the first-stage problem with a single scenario. The learned embeddings are mapped into continuous representations, and then decoded back into corresponding scenarios given the embed-

ding of the context. This design ensures that the learned representations of scenarios are correctly linked to the corresponding instances.

The resulting latent space well represents scenarios in a class of SIP instances, and we demonstrate its power with two tasks commonly used in solving SIPs. First, we leverage it for scenario reduction by finding representative scenarios via an off-the-shelf clustering algorithm, which decreases the problem complexity and meanwhile attains high-quality approximate solution. Second, we further extend the encoder with an additional sub-network to predict objective values for each scenario. Due to the generalization in the latent continuous space, we readily deploy the prediction task in semi-supervised manner with a small number of target objective values. In doing so, the latent space is augmented with the predicted objective value and hence yields more representative scenarios, which further shrinks the approximation gap to the optimal solutions of the original SIPs.

In summary, this paper aims to learn scenario representation for solving general two-stage SIPs. The main contributions are as follows:

- We firstly present a deep generative framework to learn scenario representations in two-stage SIPs. It considers dependencies of the scenarios on the deterministic context of each instance, and hence well differentiates scenarios in the latent space.
- We apply the learned representations to scenario reduction and objective prediction for SIPs. The representative scenarios derived from clustering of the representations can be used to efficiently compute high-quality solutions for SIPs. Moreover, we further shrink the approximation error by predicting objective values of scenarios via semi-supervised learning.
- We evaluate the proposed method on two SIP problems, i.e., network design problem and facility location problem. Experimental results verify the superiority of the learned scenario representation for solving SIPs. Our method significantly outperforms existing deep learning based baselines. Notably, it also generalizes well to problems with larger sizes or more scenarios.

2 Related Work

Learning based methods for SIPs. Recent works of deep learning for COPs mostly address deterministic problems, with scarce attention to uncertainty that commonly exist in practice. Though a few methods apply deep learning to improve approximate solution for two-stage SIPs, they are mostly designed for some specific problems. Nair et al. (2018) train a bit-flipping policy with reinforcement learning to improve solution iteratively, but it is limited to binary decision variables and assumes no constraints in the first-stage problem. Larsen et al. (2018) predict properties of optimal solutions to integer programs (i.e. the first-stage problem with respective single scenario) in a supervised manner. This method is only applicable when operational solutions are not needed. In contrast, Abbasi et al. (2020) predict values of decision variables for blood transshipment problem with fixed-sized instances. To reduce the complexity of SIPs, Bengio et al. (2020) aim to yield one scenario for the first-stage problem, so that the derived solution achieves a small objective value when evaluated with the original set of scenarios. This method assumes complete recourse, with any generated scenario being feasible for the second-stage problem. All the above methods attempt to reduce the expected cost of stochastic scenarios but their applications are limited by distinct assumptions or fixed problem sizes. In contrast, our method can be used for more general two-stage SIPs, and generalizes well to varying problem size or scenario quantity.

Scenario reduction for SIPs. In SIP formulation, a set of scenarios is used to model the stochastic event on top of the first-stage problem. It introduces extra complexity which may exponentially increase with its cardinality, owing to the combinatorial search space (Dyer & Stougie, 2006). Since direct optimization with massive scenarios is intractable, scenario reduction is often conducted to find a subset of scenarios that well replaces the original set with a small approximation error. Existing reduction methods mainly branches into distribution-oriented and problem-oriented paradigms. The former ones pursue a group of scenarios that closely estimate the probability distribution under the scenario space (Henrion & Römisch, 2017). The distance between the estimated and actual probability are often measured by Wasserstein distance (Rujeerapaiboon et al., 2018), polyhedral discrepancy (Henrion et al., 2008) or difference of moments (Høyland et al., 2003). These methods neglect the influence of the context on the objective function, e.g., similar scenario distributions may cause different expected cost in disparate SIP instances. Thus, there is a recent trend towards the

study of problem-oriented methods (Henrion & Römisch, 2018; Fairbrother et al., 2019; Keutchayan et al., 2020; 2021), which attempts to involve problem-specific properties in scenario reduction. For example, Keutchayan et al. (2021) propose to find K scenarios with their objective values separately approximating the expected objective value in each of the K scenario subsets. Most the above methods are mainly for theoretical proof on stochastic programs (without the restriction of integers). In contrast, we firstly propose a deep learning method for SIPs to learn scenario representation, which can be used for both scenario reduction and objective prediction.

3 PRELIMINARIES

3.1 TWO-STAGE STOCHASTIC INTEGER PROGRAMS

As an optimization problem, SIP is usually characterized by both the uncertain parameters that potentially follow certain probability distributions and the discrete solution space due to the integer restrictions. It is commonly described by a two-stage formulation as below:

$$\min_x \quad \mu^\top x + \mathbb{E}_\omega[Q(x,\omega)] \tag{1}$$

$$\text{s.t.} \quad Ax \leq b, \tag{2}$$

$$x \in \mathbb{R}^{n_1-p_1} \times \mathbb{Z}^{p_1}, \tag{3}$$

where $Q(x,\omega) := \min_y\{q_\omega^\top y | W_\omega y \leq h_\omega - T_\omega x; y \in \mathbb{R}^{n_2-p_2} \times \mathbb{Z}^{p_2}\}$.

In particular, Eq. (1), (2) and (3) prescribe the first-stage problem with $x \in \mathbb{R}^{n_1}$ being the decision variable, where $n_1 \geq p_1$; $Q(x,\omega)$ prescribes the second-stage problem, with $y \in \mathbb{R}^{n_2}$ being the decision variable, where $n_2 \geq p_2$. We refer to the group of *static* parameters $\mu \in \mathbb{R}^{n_1}$, $A \in \mathbb{R}^{m_1 \times n_1}$ and $b \in \mathbb{R}^{m_1}$ in the first stage as the *context*, and assume that the group of *uncertain* parameters $q_\omega \in \mathbb{R}^{n_2}$, $W_\omega \in \mathbb{R}^{m_2 \times n_2}$, $T_\omega \in \mathbb{R}^{m_2 \times n_1}$ and $h_\omega \in \mathbb{R}^{m_2}$ follows a distribution $\mathcal{P}$. The primary method to solve the above SIP is sample average approximation (Kleywegt et al., 2002), which converts it into a mixed integer program (MIP) by Monte Carlo simulation and in turn optimizes the following objective:

$$\min_x \quad \mu^\top x + \frac{1}{N}\sum\nolimits_{i=1}^{N} Q(x,\omega_i), \tag{4}$$

where $\{\omega_i\}_{i=1}^N$ is the set of *scenarios*, i.e., realizations of uncertain parameters. To well fit the distribution $\mathcal{P}$, a large cardinality of scenario set is often required, which may cause intractable MIP for a given solver. In this paper, we aim to find a small number of informative representatives in $\{\omega_i\}_{i=1}^N$ by learning representations of scenarios, which is supposed to considerably boost the computation efficiency with tolerable approximation errors.

3.2 CONDITIONAL VARIATIONAL AUTOENCODER

As an unsupervised generative model, CVAE is developed on top of VAE, and it further controls the data generation process conditioned on additional random variables (Sohn et al., 2015). These conditional variables could be either class labels or certain properties of input data with specific distributions, which are engaged in the input to both encoder and decoder. Optimized with stochastic gradient variational bayes (SGVB) framework (Kingma & Welling, 2014), the evidence lower bound objective (ELBO) of CVAE on the marginal likelihood for the input data is defined as:

$$\log p_\theta(X,c) \geq \text{ELBO}(X,c) = \mathbb{E}_{q_\phi(z|X,c)}[\log p_\theta(X|z,c)] - \text{KL}[q_\phi(z|X,c)\|p(z|c)], \tag{5}$$

where X, c, z denote input, latent and conditional variables, respectively; both encoder q_ϕ and decoder p_θ are typically parameterized by deep neural networks. Rather than simple distributions of conditional variables in most existing works, in this paper we exploit CVAE to learn scenario representation in SIPs, with conditional variables derived from continuous parameters in the context. For more details of CVAE, we refer readers to (Kingma et al., 2014; Feng et al., 2021).

4 METHODOLOGY

Given a class of SIP instances $\{X_m\}_{m=1}^M$ with parameters drawn from distribution $\mathcal{D}$, we regard each instance as a 2-tuple $X_m = (D_m, \{\omega_m^i\}_{i=1}^N)$, where D_m denotes the *context* (i.e. the group

of static parameters) in the m-th instance, and ω_m^i denotes the i-th scenario (i.e. the i-th realization of uncertain parameters) of the m-th instance. We aim to learn latent representations (variables) $\{z_m^i\}_{i=1}^N$ of scenarios $\{\omega_m^i\}_{i=1}^N$ under context D_m in each instance.

4.1 CVAE FOR SCENARIO REPRESENTATION

Our CVAE comprises an encoder for *inference* process and a decoder for *generation* process, which are parameterized by deep neural networks q_ϕ and p_θ, respectively. In the generation process, the decoder approximates the posterior distribution of uncertain parameters in scenarios, given the *latent* and *conditional* variables, such that:

$$p_\theta(\omega_m^i|z_m^i, c_m) = f(\omega_m^i; z_m^i, c_m, \theta); \;\; q_\phi(c_m|D_m) = h(c_m; D_m, \phi), \tag{6}$$

where f and h denote likelihood functions (e.g. Gaussian or multinomial distributions) that are instantiated by p_θ and q_ϕ, respectively. Note that we do not take the raw context D_m as conditional variables for the decoder since they are continuous and high-dimensional. It may not only intensify the computational complexity but also overwhelm the latent variable which is typically low-dimensional. In contrast, we leverage the graph neural network (GNN) in the encoder to derive a low-dimensional conditional variable c_m from D_m. Further by preserving marginal independence between latent and conditional variables, the decoder will more effectively govern the generation of ω_m^i according to the context D_m (Kingma et al., 2014).

In the inference process, the encoder approximates the posterior distribution of latent representations (variables) of scenarios given (ω_m^i, D_m), which is expressed as:

$$q_\phi(z_m^i|\omega_m^i, D_m) = \mathcal{N}(z_m^i|\mu_\phi(\omega_m^i, D_m), \sigma_\phi^2(\omega_m^i, D_m)), \tag{7}$$

where we assume a Gaussian distribution of scenario representations with μ_ϕ and σ_ϕ being parameterized by neural networks, respectively. As aforementioned, we exploit a GNN in the encoder to embed the continuous and high-dimensional ω_m^i and D_m, which is also used by the decoder to embed the sole D_m. This parameter sharing design contributes to a fast learning of embeddings for deriving the latent and conditional variables, i.e., z_m^i and c_m. To learn disentangled representations for the two types of variables, we force the conditional independence as:

$$q_\phi(z_m^i, c_m|\omega_m^i, D_m) = q_\phi(z_m^i|\omega_m^i, D_m) q_\phi(c_m|D_m). \tag{8}$$

In summary, our CVAE hinges on two important steps, i.e., 1) we first use the encoder to attain scenario representations with respect to the context in each SIP instance; 2) we then use the decoder to reconstruct the scenarios from the latent space according to the context.

Semi-supervised CVAE. With the inferred scenario representations, we can directly conduct various downstream tasks for solving SIP, such as scenario reduction using an off-the-shelf clustering algorithm. However, it may ignore the relationship between scenarios and objective values, which is nontrivial since even similar scenarios would deliver different solutions. Therefore, we also extend CVAE to predict the objective function in a semi-supervised manner. Given that the scenarios are embedded into a continuous space by the encoder, the prediction is supposed to be well generalized (Gómez-Bombarelli et al., 2018).

Most CVAE models for semi-supervised learning predict discrete targets and regard the ground truth, e.g., class labels, as conditional variables. In contrast, we predict a continuous property of the data (i.e. the objective value) and directly approximate it through the encoder. To this end, we use CPLEX solver to collect optimal objective values $\{Y_m^i\}_{i=1}^N$ of the resulting MIP problems, which are defined by respective scenarios with the context, for only a small volume ($1\% \cdot M$) of instances in $\{X_m\}_{m=1}^M$. We refer to the underlying joint distribution of (ω_m^i, Y_m^i, D_m) as $\mathcal{D}_Y$. Then we infer the objective values with a sub-network, which processes the GNN embedding of (ω_m^i, D_m) in the encoder, such that the objective function σ is parameterized as:

$$r_\psi(Y_m^i|h_m^i) = \sigma(Y_m^i; h_m^i, \psi); \;\; q_\phi(h_m^i|\omega_m^i, D_m) = g(h_m^i; \omega_m^i, D_m, \phi), \tag{9}$$

where h_m^i denotes the GNN embedding that is derived by q_ϕ.

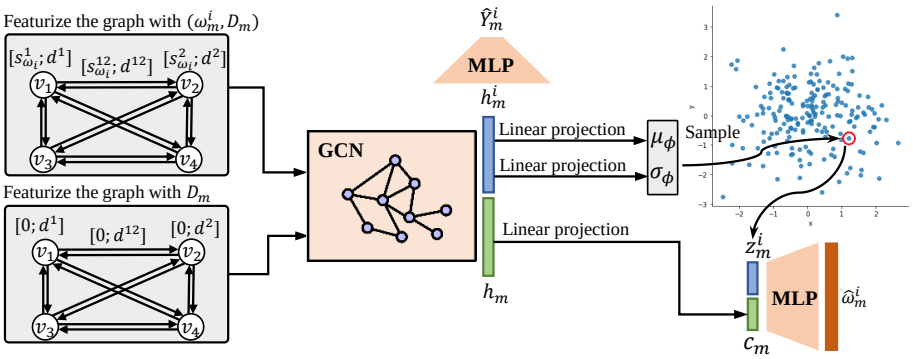

Figure 1: CVAE structure for learning scenario representations

4.2 NEURAL NETWORKS

A major challenge to parameterize CVAE for SIPs is how to derive an effective embedding of the context, since it is high-dimensional and continuous. Moreover, it is also involved in the input to both the encoder and decoder, and significantly influences the eventual scenario representation. A direct use of multi-layer perceptron (MLP) to process static parameters in D_m may fail to leverage the problem structure and cannot generalize across instances of different sizes. To overcome these limits, we exploit a GNN (as aforementioned) to embed the context D_m for attaining conditional variables in the decoder. Meanwhile, we apply the same GNN to embed each scenario with its context (ω_m^i, D_m) for attaining latent representations in the encoder. To this end, we first describe the context with scenarios on a graph.

SIP graphs. We define a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ denotes nodes with features $\mathbf{V} \in \mathbb{R}^{n \times d_v}$; $\mathcal{E} = \{e_{jk} | v_j, v_k \in \mathcal{V}\}$ denotes edges with features $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$. We formally represent the features on the j-th node as $\mathbf{v}_{\omega_i}^j = [s_{\omega_i}^j; d^j]$ ([;] means concatenation). In specific, $s_{\omega_i}^j \in \omega_m^i$ denotes a realization of uncertain parameters on the j-th node and $d^j \in D_m$ denotes static parameters on the j-th node. Similarly, the features on edge e_{jk} is represented as $\mathbf{e}_{\omega_i}^{jk} = [s_{\omega_i}^{jk}; d^{jk}]$, where $s_{\omega_i}^{jk}$ and d^{jk} denote the parameters from the i-th scenario and the context, respectively. Different from existing works that only consider varying parameters on nodes, e.g., Nazari et al. (2018), our graph representation is more generic and could be applied to a broader class of SIPs. Particularly, two examples of the graph representation on network design problem and facility location problem are introduced in Section 4.4.

On top of the SIP graphs, we can exploit GNN to derive embeddings for either the context D_m or varying scenarios with the context, i.e., (ω_m^i, D_m), $i = \{1, \ldots, N\}$. In our method, we adopt the graph convolutional network (GCN), which is a commonly used GNN variant (Kipf & Welling, 2017; Yao et al., 2019). The architecture of our GCN is similar to the one in Joshi et al. (2019), which was leveraged to compute an adjacency matrix for solving travelling salesman problem. Specifically, we extend it for general SIP problems to attain graph embeddings for D_m and (ω_m^i, D_m), denoted as h_m and h_m^i, respectively. Detailed structure of our GCN is described in Appendix A. Next, we will revisit our CVAE given the graph embeddings above.

Encoder. The encoder processes the graph embedding h_m^i with two separate linear projections to compute vectors of means and standard deviations for latent variables, respectively, i.e., $\mu_\phi(\omega_m^i, D_m)$ and $\sigma_\phi(\omega_m^i, D_m)$ in Eq. (7). Also, we linearly project the graph embedding h_m into a 2-dimensional vector, i.e., c_m in Eq. (6).

Decoder. We concatenate c_m with the latent variables z_m^i and pass them through an MLP to reconstruct the scenario. The MLP is structured by two hidden layers with a ReLU activation function, which are of 128 and 256 dimensions, respectively.

Semi-supervised learning. To echo the semi-supervised CVAE in Section 4.1, we leverage another MLP as a sub-network to process h_m^i. It comprises a 512-dimensional hidden layer with a ReLU activation function, and outputs a single value to estimate the objective value Y_m^i.

The basic structure of neural networks is illustrated in Figure 1, where $\hat{\omega}_m^i$ and $\hat{Y}_m^i$ are the reconstruction and predicted objective value for the i-th scenario, respectively. Note that we pad uncertain parameters ω_m^i with 0 to embed the sole context (in the lower grey square). In this way, the GCN could be shared to attain both graph embeddings that are used by encoder and decoder, respectively.

4.3 TRAINING & INFERENCE

The training objective is to minimize three loss functions. The first one is defined by the mean square error (MSE) between the reconstructed scenario from decoder and the original one. It is essentially used for CVAE to maximize the marginal likelihood of the data. The second one is defined by Kullback–Leibler divergence from the latent variables to the prior Gaussian distribution $p(z) = \mathcal{N}(z|0, I)$, which is used for regularization. The third one is defined by MSE between the predicted objective values and the ground truth. Formally, the objective for training is expressed as:

Algorithm 1 Training procedure

Input: encoder p_θ, decoder q_ϕ, number of epochs E, batch size B.

```
Initialize parameters θ^(0) and φ^(0)
for epoch = 1, 2, ..., E do
    shuffle the data in D and D_Y;
    for t = 1, 2, ..., ⌊|D|/B⌋ do
        Retrieve a batch of instances from D;
        get a random scenario in each instance;
        compute ELBO in Eq. (10);
        Retrieve a batch of instances from D_Y;
        compute MSE of objective value in Eq. (10);
        θ, φ, ψ ← Adam(θ, φ, ψ, ∇L(θ, φ, ψ)).
    end for
end for
```

$$\mathcal{L}(\theta, \phi, \psi) = \mathbb{E}_{\mathcal{D}}[-\text{ELBO}(X_m)] - \alpha \cdot \mathbb{E}_{\mathcal{D}_Y}[\log r_\psi(Y_m^i | h_m^i)], \tag{10}$$

where $\text{ELBO}(X_m) = \mathbb{E}_{q_\phi(z_m^i|\omega_m^i, D_m)}[\log p_\theta(\omega_m^i | z_m^i, c_m)] - \beta \cdot \text{KL}[q_\phi(z_m^i|\omega_m^i, D_m) \| p(z)]$ involves two losses; α and β are hyperparameters to balance the three losses. Following the typical training paradigm for VAE, we jointly optimize θ and ϕ via reparameterization trick and Monte Carlo approximation (Kingma & Welling, 2014). The training procedure is summarized in Algorithm 1 where we use the Adam optimizer to update the parameters (Kingma & Ba, 2015).

As shown in Algorithm 1, we randomly pick one scenario for each instance so that we can process it with its context in a batch. The resulting generalization across both the contexts and scenarios of different instances will allow for a fast training of neural networks. However, in the inference we employ the trained encoder to attain the latent representations, by processing all scenarios from a set of instances in parallel. It is realized by duplicating the context in each instance for all its scenarios. In doing so, the computational efficiency will be considerably improved for solving the instance with a large cardinality of scenarios, and thus a large amount of instances.

4.4 APPLICATIONS

We apply our method to learn scenario representation in two typical SIP problems, i.e., network design problem (NDP) and facility location problem (FLP) (Keutchayan et al., 2021). We represent them as the SIP graphs, whose features on nodes and edges including both static and uncertain parameters are listed in Table 1. For FLP, the parameters tokenized by ***F.*** and ***C.*** mean that they are peculiar to facility and customer nodes, respectively. More details about the two problems and their parameters can be found in Appendix B and C.

Table 1: Features on SIP graphs of NDP and FLP

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

5 EXPERIMENTS

We evaluate our method on NDP and FLP to demonstrate the effectiveness of the learned latent scenario representations. On the one hand, the representations in each instance can be directly used for scenario reduction by using an off-the-shelf clustering algorithm to find representatives. On the other hand, we can also predict objective values for each scenario and append them to the representations to find more informative representatives. We refer to the two paradigms as CVAE-SIP and

CVAE-SIPA, respectively. Then we compare them with other learning based methods, and assess their generalization performance across problems with varying sizes and scenario cardinalities.

5.1 SETUP

Instance generation. Our instances are generated following the way in (Keutchayan et al., 2021). Specifically, we generate NDP instances with 14 nodes including 2 source nodes, 2 target nodes and 10 transition nodes. For FLP, we designate 10 facility nodes and 20 customer nodes. We generate 200 scenarios for each instance of both problems. To evaluate the generalization performance, we generate larger instances by *doubling* and *quadrupling* the number of *transition* nodes for NDP, respectively. We do the same to both facility and customer nodes for FLP. Besides, we also double and quadruple the scenario cardinality in both problems. More details are provided in Appendix D.

Training. We collect $12,800$ instances (of normal size) for NDP and FLP, respectively, and normalize each type of parameters into $[0, 1]$ across instances. Regarding CVAE-SIP, we train our model without the sub-network (used for the prediction task) by solely optimizing the ELBO loss in Eq. (10). Regarding CVAE-SIP, we train neural networks for learning representation and objective prediction simultaneously, following the procedure in Algorithm 1. We set hyperparameters $\beta = 0.005$ and $\alpha = 100$ in Eq. (10). For both problems, CVAE-SIP and CVAE-SIPA are trained with 100 and 400 epochs, respectively. The batch size and learning rate are set to 128 and 10^{-4}. We run the training on a server using a single GeForce RTX 2080 Ti GPU.

Testing. Pertaining to CVAE-SIP, we employ the K-medoids algorithm (Park & Jun, 2009) to obtain representative scenarios based on the latent representations. We keep its default settings and only change the number of clusters (i.e. K). Given the attained K representatives, we can derive the approximation solution by reformulating Eq. (4) as:

$$\min_{x} \quad \mu^\top x + \frac{N_{\mathsf{k}}}{N} \sum\nolimits_{\mathsf{k}=1}^{K} Q(x, \omega_{\mathsf{k}}), \tag{11}$$

where ω_{k} means the representative in the k-th cluster and the number of intra-cluster population equals to N_{k}. Accordingly, the approximation error is defined as the gap between the objective values in Eq. (11) and Eq. (4). Pertaining to CVAE-SIPA, we infer both the latent representations and objective values of scenarios, which are concatenated as augmented representations to attain the representatives. To reduce the influence by the potential concurrent processes, we conduct the evaluation on a different desktop computer, which uses a single Nvidia Quadro P4000 GPU with a Xeon W-2133 CPU@3.60 GHz.

5.2 COMPARISON ANALYSIS

We compare with four baselines, i.e., 1) CPLEX IBM (2017), with default settings to compute optimal solutions. It is also used to solve the approximation problem with representative scenarios. 2) K-medoids, the same clustering algorithm as used in our method. However, we apply it to directly cluster the original scenarios rather than the learned low-dimensional representations. 3) Scenario-M, a supervised method with hand-crafted features to predict a scenario for solving a variant of FLP (Bengio et al., 2020). 4) Solution-M[1], a supervised method with raw features to predict the first-stage decision variables for solving blood transshipment problem (Abbasi et al., 2020).

We evaluate all methods on 200 instances of normal size for NDP and FLP, respectively. We set K = 3, 5, 10 in K-medoids for both problems so that we could access the performance against the number of representatives. Note that the computation by CPLEX, Scenario-M and Solution-M is independent of K. The results are displayed in Table 2, where we record the average of objective values, approximation errors and computational time over all instances, respectively. As shown, both CVAE-SIP and CVAE-SIPA consistently outperform K-medoids in all cases. Though K-medoids can also improve the solution quality as K grows, our methods efficiently attain solutions with much smaller errors. It implies that we achieve more informative representatives with the learned low-dimensional representations, which help deliver high-quality solutions. On the other hand, our methods significantly outperform Scenario-M for the three respective values of K, and exceed Solution-M for K = 5 and 10. This suggests that the representatives found by our methods are able

[1]The training details of Scenario-M and Solution-M are summarized in Appendix E.

Table 2: Comparison with baselines

	NDP (14; 200)									FLP (30; 200)								
	K=3			K=5			K=10			K=3			K=5			K=10		
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	1348.24	1.12	0.3s	976.47	0.54	0.3s	761.18	0.19	0.6s	2405.27	9.40	0.1s	2390.06	9.06	0.2s	1344.39	4.57	0.3s
CVAE-SIP	1309.79	1.06	0.6s	930.73	0.48	0.7s	734.24	0.15	0.9s	1384.53	4.91	0.6s	929.61	2.87	0.7s	482.70	0.99	0.7s
CVAE-SIPA	959.96	0.51	0.6s	**769.70**	**0.24**	0.6s	**687.68**	**0.08**	0.8s	1561.07	5.62	0.5s	**709.08**	**1.71**	0.6s	**381.41**	**0.58**	0.7s

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

Table 3: Generalization to large-scale problems

	NDP (24; 200)									FLP (60; 200)								
	K=3			K=5			K=10			K=3			K=5			K=10		
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	1300.57	1.16	0.5s	981.94	0.63	0.8s	723.62	0.20	1s	4571.44	12.75	1s	3549.25	9.72	1s	1549.53	3.74	4s
CVAE-SIP	1313.05	1.18	1s	871.73	0.45	2s	696.22	0.16	2s	2224.56	5.52	2s	1083.12	2.17	3s	748.59	1.21	5s
CVAE-SIPA	**1008.02**	**0.68**	1s	**809.65**	**0.34**	2s	**637.72**	**0.06**	2s	**1764.29**	**4.22**	2s	**974.17**	**1.86**	3s	**746.02**	**1.18**	5s

	NDP (44; 200)									FLP (120; 200)								
	K=3			K=5			K=10			K=3			K=5			K=10		
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	1310.48	0.99	1s	803.99	0.38	3s	652.78	0.12	7s	12748.67	24.87	3s	6675.13	12.56	6s	2803.77	4.86	21s
CVAE-SIP	1031.31	0.64	5s	1079.80	0.84	5s	**615.99**	**0.06**	8s	4865.53	8.99	10s	**1585.53**	**2.27**	12s	985.45	0.99	35s
CVAE-SIPA	**1010.73**	**0.62**	5s	**682.35**	**0.17**	5s	627.62	0.08	8s	**3353.57**	**5.92**	12s	2051.45	3.13	16s	**910.50**	**0.84**	33s

to approximate the original problem more accurately than the approximation with a single scenario or solution. Furthermore, we observe that CVAE-SIPA is generally superior to CVAE-SIP, which indicates that the predicted objective values could effectively enrich the scenario representations for achieving more informative representatives and thus better solutions. Last but not least, all the learning based methods consume fairly short time compared to the sole CPLEX. This advantage is more evident as the problems become more complex, which is revealed in the following experiments.

5.3 Generalization across problem sizes

A desirable generalization to larger problems is necessary for learning based methods, since practical problems are often large-scale. To demonstrate such capability of our method, we apply the previously trained encoder to directly infer another 50 larger instances (of two levels) for both NDP and FLP, respectively. The instances are generated following the operation of *doubling* and *quadrupling* decribed in Section 5.1. The baselines are also directly applied to these larger instances except Solution-M since its output dimension (i.e. number of decision variables) is fixed only for the problem of normal size. Similarly, Scenario-M is not applicable to FLP where the scenario dimension (i.e. number of customer nodes) changes with problem sizes. We display all results in Table 3, where the upper and lower half contains the results on *doubling* and *quadrupling* instances, respectively. As shown, CVAE-SIP performs superior to K-medoids and Scenario-M in most cases except NDP (24; 200) with K=3 and NDP (44; 200) with K=5. On the other hand, CVAE-SIPA generally achieves the smallest errors but performs inferior to CVAE-SIP on NDP (44; 200) with K=10 and FLP (120; 200) with K=5. This might stem from the degenerated accuracy of objective prediction, which attains solutions with relatively large penalties on some instances. Notably, the computational time by CPLEX rises drastically on larger problems, in comparison with those in Table 2. Such low efficiency might not be acceptable in practice, especially where real-time decisions are required for handling a number of problem instances simultaneously. In contrast, our methods deliver solutions with small penalties in short runtime. Moreover, our methods can further reduce the error on FLP with a few more representative scenarios, which are demonstrated in Appendix F.

Table 4: Generalization to large scenario cardinalities

	NDP (14; 400)									FLP (30; 400)								
	K=3			K=5			K=10			K=3			K=5			K=10		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	634.50	0.00	1m	634.50	0.00	1m	634.50	0.00	1m	244.63	0.00	2m	244.63	0.00	2m	244.63	0.00	2m
Scenario-M	2266.88	2.58	1s	2266.88	2.58	1s	2266.88	2.58	1s	5260.62	12.79	2s	5260.62	12.79	2s	5260.62	12.79	2s
Solution-M	**791.95**	**0.26**	0.7s	**791.95**	**0.26**	0.7s	791.95	0.26	0.7s	1215.86	5.37	0.5s	1215.86	5.37	0.5s	1215.86	5.37	0.5s
K-medoids	1148.86	0.81	0.2s	972.69	0.53	0.3s	730.18	0.15	0.5s	2655.03	9.77	0.2s	1848.21	6.12	0.2s	734.44	2.00	0.4s
CVAE-SIP	1094.80	0.72	0.8s	887.76	0.41	0.8s	711.63	0.12	1s	**923.42**	**2.74**	1s	**701.73**	**1.83**	1s	468.47	0.86	1s
CVAE-SIPA	1032.06	0.62	0.8s	881.05	0.38	0.8s	**676.99**	**0.06**	1s	1599.43	5.77	1s	723.75	1.89	1s	**439.08**	**0.77**	1s

	NDP (14; 800)									FLP (30; 800)								
	K=3			K=5			K=10			K=3			K=5			K=10		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	633.34	0.00	5m	633.34	0.00	5m	633.34	0.00	5m	239.45	0.00	7m	239.45	0.00	7m	239.45	0.00	7m
Scenario-M	2137.65	2.36	0.7s	2137.65	2.36	0.7s	2137.65	2.36	0.7s	3807.41	9.93	2s	3807.41	9.93	2s	3807.41	9.93	2s
Solution-M	**815.13**	**0.28**	0.8s	815.13	0.28	0.8s	815.13	0.28	0.8s	1082.62	3.76	0.7s	1082.62	3.76	0.7s	1082.62	3.76	0.7s
K-medoids	1302.71	1.05	0.2s	896.35	0.42	0.3s	770.99	0.20	0.7s	2025.77	7.61	0.2s	1174.66	4.18	0.2s	1091.77	3.34	0.3s
CVAE-SIP	1600.19	1.52	1s	900.59	0.42	1s	**658.40**	**0.03**	2s	1199.26	4.39	2s	**600.28**	**1.36**	2s	**351.18**	**0.49**	2s
CVAE-SIPA	982.24	0.56	1s	**761.84**	**0.20**	1s	684.33	0.07	2s	**1039.90**	**3.30**	2s	990.76	2.91	2s	420.95	0.77	2s

5.4 Generalization across scenario cardinalities

From a practical view, the cardinality of scenarios in SIP problems might be changed due to certain reasons, e.g., new observations are added or outdated ones are removed (Issac & Campbell, 2017). Therefore, a learning based method for SIPs should be able to handle this varying dimension via generalization. To this end, we evaluate our method and baselines on 50 instances of NDP and FLP with *doubling* and *quadrupling* scenarios, respectively. The results in Table 4 show that our methods can still achieve the smallest errors across most settings, except three cases with small K on NDP, where Solution-M performs better. It implies that our methods are less sensitive to the scenario cardinality. We also observe that the advantage of CVAE-SIPA over CVAE-SIP turns weak, especially on FLP. This might be caused by frequent objective predictions for scenarios in CVAE-SIPA, which potentially leads to penalties on more instances. Since our methods process all scenarios in parallel, they solve instances with scenarios up to 800 in about 2 seconds, which is significantly faster than CPLEX. Additionally, we also test with scenarios generated from different distributions to verify the cross-distribution generalization ability. The details are provided in Appendix G.

5.5 Objective prediction

We further demonstrate the efficacy of the semi-supervised learning for objective prediction. The trained model is directly adopted to infer objective values of scenarios in NDP and FLP instances. The results are gathered in Figure 4 and 3 in Appendix H, which show our prediction is fairly accurate for both problems even when generalizing to larger problem sizes or scenario cardinalities. It well verifies the effectiveness of our semi-supervised design, which only needs to solve a small volume (1%) of training instances. In addition, the sample efficiency is also enhanced through the generalization over the latent space. More details can be found in Appendix H.

6 Conclusions and future work

In this paper, we present a deep learning method to solve two-stage SIPs. We leverage CVAE to learn latent representations of stochastic scenarios conditioned on the context of a SIP instance. With the learned representations, we conduct two downstream tasks for solving SIP, i.e., scenario reduction and objective prediction. The evaluation on two classic SIP problems has shown that our method is able to find less yet informative representatives for improving both solution quality and computation efficiency. Additional studies also verify that our method can be well generalized to larger problem sizes, larger cardinalities of scenarios and different distributions of scenarios. In future, we will apply our method to other SIP problems, e.g., stochastic routing, and also attempt to automate the decision on the optimal number of representatives for scenario reduction.

REFERENCES

Babak Abbasi, Toktam Babaei, Zahra Hosseinifard, Kate Smith-Miles, and Maryam Dehghani. Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management. *Computers & Operations Research*, 119:104941, 2020.

Yoshua Bengio, Emma Frejinger, Andrea Lodi, Rahul Patel, and Sriram Sankaranarayanan. A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 99–111. Springer, 2020.

Sergio Bruno, Shabbir Ahmed, Alexander Shapiro, and Alexandre Street. Risk neutral and risk averse approaches to multistage renewable investment planning under uncertainty. *European Journal of Operational Research*, 250(3):979–989, 2016.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *the 33rd Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 6278–6289, 2019.

Mary Dillon, Fabricio Oliveira, and Babak Abbasi. A two-stage stochastic programming model for inventory management in the blood supply chain. *International Journal of Production Economics*, 187:27–41, 2017.

Martin Dyer and Leen Stougie. Computational complexity of stochastic programming problems. *mathematical programming*, 106(3):423–432, 2006.

Jamie Fairbrother, Amanda Turner, and Stein W Wallace. Problem-driven scenario generation: an analytical approach for stochastic programs with tail risk measure. *Mathematical Programming*, pp. 1–42, 2019.

Hao-Zhe Feng, Kezhi Kong, Minghao Chen, Tianye Zhang, Minfeng Zhu, and Wei Chen. Shot-vae: Semi-supervised deep generative models with label-aware elbo approximations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*, pp. 7413–7421, 2021.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NIPS)*, 2019.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

René Henrion and Werner Römisch. *Optimal scenario generation and reduction in stochastic programming*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät ..., 2017.

Réne Henrion and Werner Römisch. Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming*, pp. 1–23, 2018.

René Henrion, Christian Küchler, and Werner Römisch. Discrepancy distances and scenario reduction in two-stage stochastic mixed-integer programming. *Journal of Industrial & Management Optimization*, 4(2):363, 2008.

Kjetil Høyland, Michal Kaut, and Stein W Wallace. A heuristic for moment-matching scenario generation. *Computational optimization and applications*, 24(2):169–185, 2003.

IBM(2017). Ibm ilog cplex 12.10 user manual ibm crop. 2017.

Preethi Issac and Ann Melissa Campbell. Shortest path problem with arc failure scenarios. *EURO Journal on Transportation and Logistics*, 6(2):139–163, 2017.

Haifa Jammeli, Majdi Argoubi, and Hatem Masri. A bi-objective stochastic programming model for the household waste collection and transportation problem: case of the city of sousse. *Operational Research*, pp. 1–27, 2019.

Waldy Joe and Hoong Chuin Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pp. 394–402, 2020.

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. In *INFORMS Annual Meeting*, 2019.

Julien Keutchayan, David Munger, and Michel Gendreau. On the scenario-tree optimal-value error for stochastic programming problems. *Mathematics of Operations Research*, 45(4):1572–1595, 2020.

Julien Keutchayan, Janosch Ortmann, and Walter Rei. Problem-driven scenario clustering in stochastic optimization. *arXiv preprint arXiv:2106.11717*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *the 28th Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 3581–3589, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *the 5th International Conference on Learning Representations (ICLR)*, 2017.

Kim-Manuel Klein. About the complexity of two-stage stochastic ips. *Mathematical Programming*, pp. 1–19, 2021.

Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *the 7th International Conference on Learning Representations (ICLR)*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *the 34th Conference on Advances in Neural Information Processing Systems (NIPS)*, 2020.

Eric Larsen, Sébastien Lachapelle, Yoshua Bengio, Emma Frejinger, Simon Lacoste-Julien, and Andrea Lodi. Predicting solution summaries to integer linear programs under imperfect information with machine learning. *arXiv preprint arXiv:1807.11876*, 2018.

Mengyuan Lee, Seyyedali Hosseinalipour, Christopher Greg Brinton, Guanding Yu, and Huaiyu Dai. A fast graph neural network-based method for winner determination in multi-unit combinatorial auctions. *IEEE Transactions on Cloud Computing*, 2020.

Vinod Nair, Dj Dvijotham, Iain Dunning, and Oriol Vinyals. Learning fast optimizers for contextual stochastic integer programs. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 591–600, 2018.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *the 32nd Conference on Neural Information Processing Systems (NIPS)*, pp. 9839–9849, 2018.

Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

Napat Rujeerapaiboon, Kilian Schindler, Daniel Kuhn, and Wolfram Wiesemann. Scenario reduction revisited: Fundamental limits and guarantees. *Mathematical Programming*, pp. 1–36, 2018.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *the 29th Conference on Advances in Neural Information Processing Systems (NIPS)*, 28:3483–3491, 2015.

Stein W Wallace and William T Ziemba. *Applications of stochastic programming*. SIAM, 2005.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*, pp. 7370–7377, 2019.

Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. In *the 34th Conference on Advances in Neural Information Processing Systems (NIPS)*, volume 33, pp. 1621–1632, 2020.

A GRAPH CONVOLUTIONAL NETWORK

Given a complete SIP graph as defined in Section 4.2, we first process the features of nodes and edges by separate linear transformations as below:

$$\mathbf{V}^0 = \mathbf{V}\mathbf{W}_v^0 + \mathbf{b}_v; \quad \mathbf{E}^0 = \mathbf{E}\mathbf{W}_e^0 + \mathbf{b}_e, \tag{12}$$

where $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ and $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ are the raw features (i.e. parameters in Table 1) on nodes and edges, respectively; $\mathbf{W}_v^0 \in \mathbb{R}^{d_v \times 128}$ and $\mathbf{W}_e^0 \in \mathbb{R}^{d_e \times 128}$ are trainable weights with corresponding bias vectors $\mathbf{b}_v \in \mathbb{R}^{1 \times 128}$ and $\mathbf{b}_e \in \mathbb{R}^{1 \times 128}$. Note that we also use different linear transformations for heterogeneous nodes or edges. For example, the location nodes of facilities and customers are processed separately.

Next, we advance the node and edge embeddings by the graph convolution layer for L iterations, and each iteration has its own parameters, which are expressed as:

$$\begin{aligned} \mathbf{v}_j^{l+1} &= \mathbf{v}_j^l + \text{PReLU}\Big(\text{BN}\big(\mathbf{v}_j^l \mathbf{W}_1^l + \sum_{k}^{\mathbf{e}_{jk} \in \mathcal{V}} \eta_{jk}^l \odot \mathbf{v}_j^l \mathbf{W}_2^l\big)\Big), \;\; \eta_{jk}^l = \frac{\sigma(\mathbf{e}_{jk}^l)}{\sum_k^{\mathbf{e}_{jk} \in \mathcal{V}} \sigma(\mathbf{e}_{jk}^l) + \epsilon} \\ \mathbf{e}_{jk}^{l+1} &= \mathbf{e}_{jk}^l + \text{PReLU}\Big(\text{BN}\big(\mathbf{e}_{jk}^l \mathbf{W}_3^l + \mathbf{v}_j^l \mathbf{W}_4^l + \mathbf{v}_k^l \mathbf{W}_5^l\big)\Big), \;\; l = 0, \ldots, L-1 \end{aligned} \tag{13}$$

where $\mathbf{W}_1^l, \ldots, \mathbf{W}_5^l \in \mathbb{R}^{128 \times 128}$; $\odot$ means the element-wise product; σ is the sigmoid activation function; ϵ is a small value to avoid numerical error. Initially, the node embedding $\mathbf{v}_j^0$ is the j-th row of matrix $\mathbf{V}^0$ and edge embedding $\mathbf{e}_{jk}^0$ is the element at the j-th row and k-th column of matrix $\mathbf{E}^0$. The above graph layer is similar to the one in Joshi et al. (2019) except that we use parametric rectified linear unit with its learnable parameter being 0.25.

After L (L=10) iterations of the above convolution, the node and edge embeddings are represented as $\{\mathbf{v}_j^L\}_{j=1}^n$ and $\{\mathbf{e}_{jk}^L\}_{j,k=1}^n$, respectively. We attain the graph embedding by the mean pooling over all the node embeddings. We neglect edge embeddings considering that their information has already been involved in node embeddings, after multiple convolutions.

Parameter sharing The graph embedding network based on Eq. (12) and (13) is able to process the sole context D_m with its static parameters on the graph. Also it can process the context with any scenario in the instance, by featurizing the graph with both static parameters in D_m and a scenario ω_m^i. To foster the learning efficiency, we use a single network to process these two featurizations. It is achieved by padding the uncertain parameters with zeros when we only process the context. We represent the resulting graph embeddings by h_m and h_m^i, which are further used to produce conditional and latent variables and meanwhile predict objective values, as stated in Section 4.2.

B NETWORK DESIGN PROBLEM

Given a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each edge $e_{jk} \in \mathcal{E}$ points from v_j to v_k where $v_j, v_k \in \mathcal{V}$. The features on an edge include its capacity q_{jk}, opening cost o_{jk} and transportation cost t_{jkc} for one unit of commodity $c \in C$.

Scenarios The uncertainty exists in the demands on each node. In specific, the parameter d_{vc}^i denotes the demand of commodity c on node v, in the i-th scenario.

Decision variables The decision variable in the first stage is denoted by x_{jk}. Specifically, $x_{jk} = 1$ means that the edge e_{jk} is open to transport the commodities, otherwise $x_{jk} = 0$. The decision variable in the second stage is denoted by y_{jkc}^i, which means the units of commodity c transported along the edge e_{jk} in the i-th scenario.

The objective of NDP is to decide the edges to be opened, which minimizes the total opening cost and expected transportation cost. The corresponding two-stage SIP formulation is expressed as:

$$\min \quad \sum_{e_{jk}\in\mathcal{E}} o_{jk}x_{jk} + \frac{1}{N}\sum_{i=1}^{N}\sum_{c\in C}\sum_{e_{jk}\in\mathcal{E}} t_{jkc}y^i_{jkc} \tag{14}$$

$$\text{s.t.} \quad \sum_{v_j=v}^{e_{jk}\in\mathcal{E}} y^i_{jkc} - \sum_{v_k=v}^{e_{jk}\in\mathcal{E}} y^i_{jkc} = d^i_{vc}, \qquad \forall (v,c,i)\in\mathcal{V}\times C\times\{1,\dots,N\} \tag{15}$$

$$\sum_{c\in C} y^i_{jkc} \le q_{jk}x_{jk}, \qquad \forall (v_j,v_k,i)\in\mathcal{V}\times\mathcal{V}\times\{1,\dots,N\} \tag{16}$$

$$x_{jk}\in\{0,1\}; y^i_{jkc}\in[0,\infty). \tag{17}$$

C FACILITY LOCATION PROBLEM

The facility location problem comprises F and C locations for facilities and customers, respectively. Each facility location is featured by an opening cost o_f, $f\in\{1,\dots,F\}$ and a facility has q units of resources that can be provided to customers. We assume that the number of facilities to be opened is not larger than v. For each customer $c\in\{1,\dots,C\}$, it consumes q_{cf} units of resources if it is served by the facility at the location f, and the induced service cost is denoted by s_{cf}.

Scenarios The uncertainty exists in the presence of customers. In specific, the parameter h^i_c refers to the presence of the customer c in the i-th scenario, i.e., $h^i_c = 1$ means that the customer c is present to be served in scenario i, otherwise $h^i_c = 0$.

Decision variables The decision variable in the first stage is denoted by x_f. Specifically, $x_f = 1$ means that a facility is open at location f to provide services, otherwise $x_f = 0$. The second-stage decision variables include y^i_{cf} and z^i_f. In particular, $y^i_{cf} = 1$ means that the customer c is served by facility at location f in the i-th scenario, otherwise $y^i_{cf} = 0$. Since the capacity of a facility may not meet the total demand of its customers, z^i_f determines the quantity of additional resources required at facility location f, in the i-th scenario.

The objective is to decide the locations to build the facilities, so as to minimize the total opening cost and the expected service cost including the potential penalty. The two-stage SIP formulation of the facility location problem is expressed as:

$$\min \quad \sum_{f\in F} o_f x_f + \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{c\in C}\sum_{f\in F} s_{cf}y^i_{cf} + \sum_{f\in F} b_f z^i_f\right) \tag{18}$$

$$\text{s.t.} \quad \sum_{f\in F} x_f \le v, \tag{19}$$

$$\sum_{c\in C} q_{cf}y^i_{cf} \le qx_f + z^i_f, \qquad \forall (f,i)\in F\times\{1,\dots,N\} \tag{20}$$

$$z^i_f \le Mx_f, \qquad \forall (f,i)\in F\times\{1,\dots,N\} \tag{21}$$

$$\sum_{f\in F} y^i_{cf} = h^i_c, \qquad \forall (c,i)\in C\times\{1,\dots,N\} \tag{22}$$

$$x_f, y^i_{cf}\in\{0,1\}; z^i_f\in[0,\infty), \tag{23}$$

where b_f denotes the penalty of one additional unit of resource at location f; M is a large value that constrains z^i_f to 0 if $x_f = 0$, so as to ensure that the facility f will not be penalized if it is not open.

D DETAILS OF INSTANCE GENERATION

We generate NDP instances with 14 nodes, in which the commodities on two source nodes are transported to another two target nodes to meet their demands. The remaining nodes are only for

transition without demands. We set two types of commodities and their quantities are uniformly sampled from the interval $[5, 15]$ in each scenario. For each edge, we uniformly sample the opening cost, shipping cost and capacity from $[3, 11]$, $[5, 11]$ and $[10, 41]$, respectively. A penalty 1000 will be triggered for each source node if its commodities cannot be fully transported due to the capacity limit of opened edges. For FLP, we consider 10 facility nodes and 20 customer nodes, with their coordinates uniformly sampled from the unit square $[0, 1] \times [0, 1]$. The opening cost and capacity for each facility are uniformly sampled from $[40, 81]$ and $[30, 60]$. We generate binary values through Bernoulli sampling for customer nodes to indicate their presence in a scenario, with probabilities uniformly sampled from $[0.4, 0.6]$ for each node. The penalty is also set to 1000 for each facility node if it cannot meet the demands. We generate 200 scenarios for each instance of both problems. To test the generalization performance, we generate larger instances by *doubling* and *quadrupling* the number of *transition* nodes for NDP, respectively. We do the same to both facility and customer nodes for FLP. Regarding the scenario, we also double and quadruple its cardinality in both problems.

E TRAINING DETAILS FOR THE TWO BASELINES

Scenario-M This method aims to generate one scenario, which can result in a good approximate solution. It is not easy since for each instance, sufficient scenarios need to be evaluated to attain the target scenario for supervised learning. To avoid much computational burden, Bengio et al. (2020) compute the average of scenarios as the initial one and then iteratively improve it to derive a similar solution as the optimal one. However, the improvement heuristic used cannot guarantee that the attained one exists in the original set of scenarios. To tackle this issue, we adapt the method to track the improvement along existing scenarios. Specifically, we find the 20 scenarios nearest to the average and compute the MIPs defined by each of them with the context. We determine the target scenario by picking the one with the most similar solution to the optimal one. This method could be used for both NDP and FLP.

We conduct the same featurization as described in Appendix A.4 in (Bengio et al., 2020), and use both linear regression (LR) and artificial neural network (ANN) with MSE loss. In addition, we also use the same GCN in our method to directly process raw parameters in SIP instances. Specifically, we only input the context with the average of scenarios to GCN, and the graph embedding is linearly projected into output to estimate scenarios. The reported results in the main paper are delivered by GCN, which we find outperforms LR and ANN in most cases.

Solution-M Following (Abbasi et al., 2020), we directly predict the first-stage solution in both NDP and FLP. Despite the machine learning algorithms listed in the paper, we use the same GCN in our method for training. It is more powerful to process the raw parameters under the graph representation. In our cases, the raw parameters are the static parameters in the context for both NDP and FLP. We take them as input into GCN and the output dimension is same as the number of first-stage decision variables. If there is no static parameters on nodes or edges, we use 0 for them to run GCN as usual.

For both supervised methods, we use 12,800 instances (of normal size) in the training with the number of epochs 100, batch size 64 and learning rate 0.001. We test them with diverse instances as described in corresponding experiments in the main paper.

F ADDITIONAL RESULTS ON LARGE PROBLEMS

We further evaluate our methods on FLP problems of larger sizes. We follow the experimental setting in Section 5.3 and apply the trained model to solve 50 instances from FLP (60; 200) and FLP (120; 200). We set K from 1 to 20 and thus attain solutions with varying numbers of representative scenarios. The curve of average approximation errors is plotted in Figure 2. We observe that both CVAE-SIP and CVAE-SIPA persistently reduce errors towards 0 as K grows. It means that our methods are consistently attaining more effective representatives from scenarios, which help deliver better solutions. Meanwhile, CVAE-SIP and CVAE-SIPA outperform K-medoids with different K on both problems, especially on FLP (120; 200). It again verifies the merit of the learned representations for solving SIP problems. The comparison between CVAE-SIP and CVAE-SIPA implies that

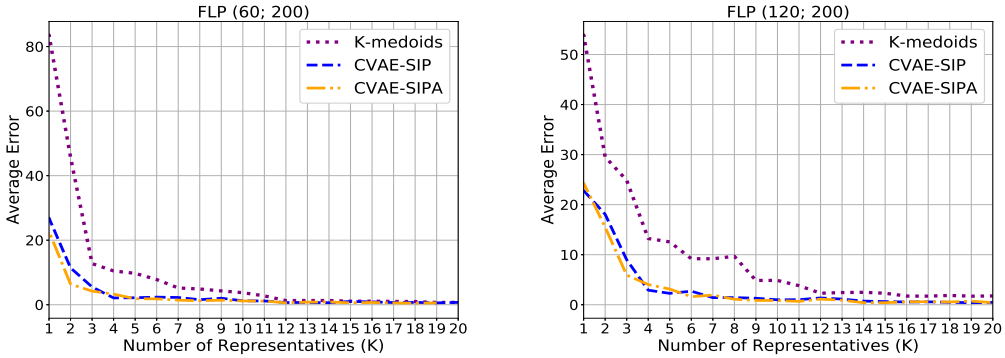

Figure 2: Errors with different values of K

though CVAE-SIPA is superior to CVAE-SIP at the early stage of the curve, i.e., K=1,. . .3, they are on par with each other when more representatives are used eventually.

G GENERALIZATION ACROSS DISTRIBUTIONS

We further test the generalization performance of our method on NDP with scenarios sampling from the Normal distribution and Binomial distribution. For the former, we set the expectation and standard deviation to 10 and 2, respectively. For the latter, we set the number of runs and the probability to 20 and 0.5. The results are summarized in Table 5. We observe that our methods consistently outperform other learning based approaches with scenarios from both distributions. When K=10, both CVAE-SIP and CVAE-SIPA achieve near-optimal solutions with slight approximation errors. On the other hand, it can be found that the results delivered by CVAE-SIPA are generally superior or comparable to those by CVAE-SIP, suggesting a better cross-distribution generalization.

Table 5: Generalization results across distributions

	NDP (14; 200); Normal									NDP (30; 200); Binomial								
	K=3			K=5			K=10			K=3			K=5			K=10		
Method	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time	Obj.	Error	Time
CPLEX	513.82	0.00	16s	513.82	0.00	16s	513.82	0.00	16s	510.08	0.00	20s	510.08	0.00	20s	510.08	0.00	20s
Scenario-M	1039.48	0.93	0.8s	1039.48	0.93	0.8s	1039.48	0.93	0.8s	1271.23	1.21	0.6s	1271.23	1.21	0.6s	1271.23	1.21	0.6s
Solution-M	678.74	0.34	0.8s	678.74	0.34	0.8s	678.74	0.34	0.8s	729.81	0.51	0.2s	729.81	0.51	0.2s	729.81	0.51	0.2s
K-medoids	662.48	0.30	0.2s	585.66	0.14	0.3s	546.51	0.06	0.5s	684.71	0.34	0.3s	623.76	0.22	0.3	583.25	0.14	0.5s
CVAE-SIP	622.98	0.23	0.6s	**548.45**	**0.07**	0.6s	**527.59**	**0.03**	0.9s	695.60	0.36	0.6s	**592.40**	**0.16**	0.6	555.35	0.09	0.9s
CVAE-SIPA	**562.87**	**0.11**	0.5s	551.94	0.07	0.6s	531.94	0.04	0.82	**622.91**	**0.22**	0.5s	592.91	0.16	0.6	**540.54**	**0.06**	0.9s

H ADDITIONAL RESULTS ON OBJECTIVE PREDICTION

We first evaluate our trained model for the prediction task in three FLP instances, which are randomly generated from FLP (30; 200), FLP (120; 200) and FLP (30; 800), respectively. For each instance, we draw a scatter plot of the 2-dimensional latent representations for scenarios and represent the normalized objective values using different colors. The results are shown in the upper half of Figure 3, and the corresponding ground truth (i.e. optimal solution values attained by CPLEX) are displayed in the lower half. It is clear that the prediction is fairly accurate with almost the same patterns as the target values in the ground truth, even for the larger problem size or scenario cardinality. It indicates that our design for semi-supervised learning is effective and the trained model exhibits a desirable generalization ability.

We further evaluate the objective prediction on NDP. Specifically, the trained model is used to predict objective values for the scenarios from three instances, which are randomly generated from NDP (14; 200), NDP (44; 200) and NDP (14; 800), respectively. From the results in Figure 4, we observe that our method can also predict the objective values accurately for NDP. Especially, the pattern of

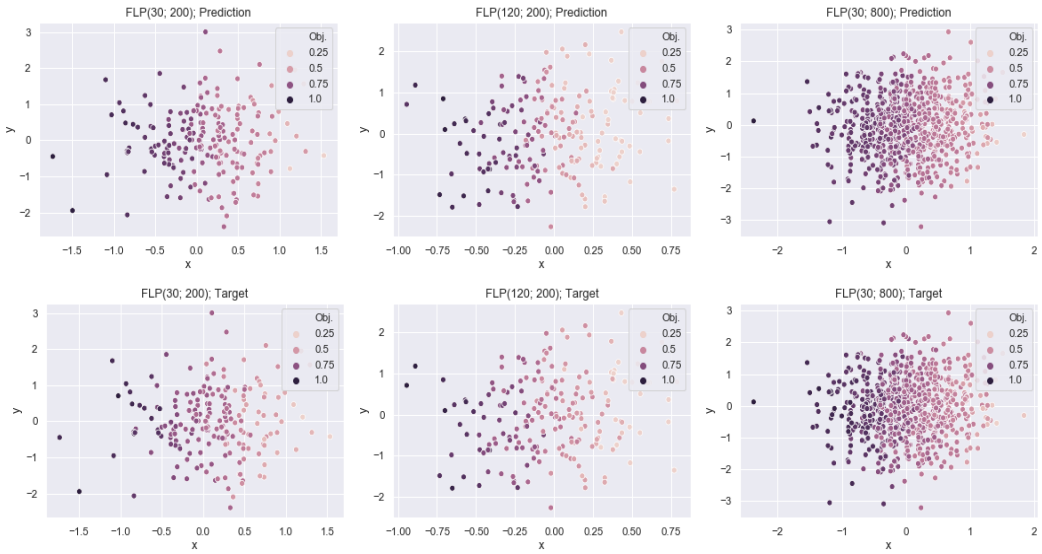

Figure 3: Objective prediction against latent representations (FLP)

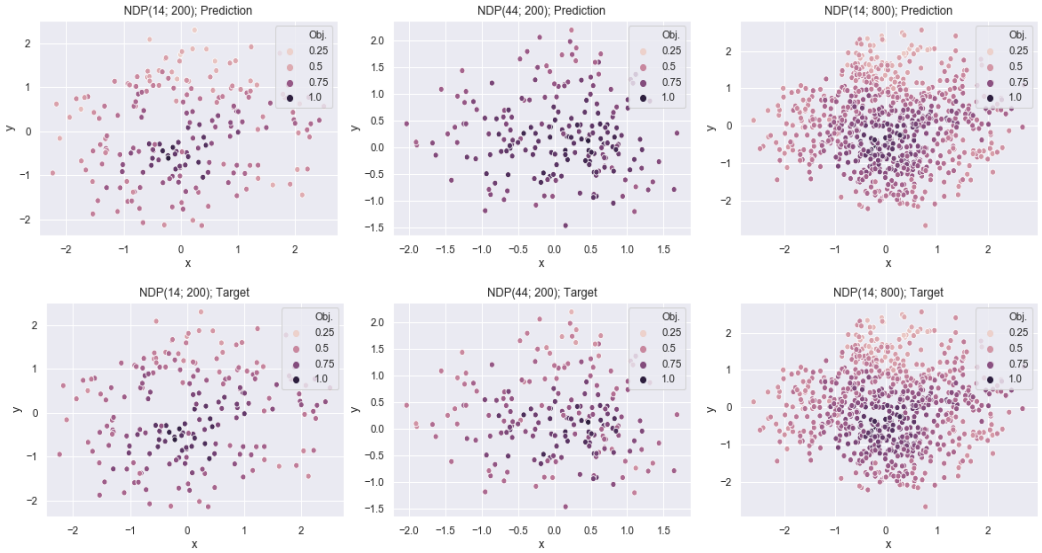

Figure 4: Objective prediction against latent representations (NDP)

predicted values for NDP (14; 800) is almost indistinguishable from the one by ground truth. It reveals that our design for the semi-supervised learning is effective across distinct SIPs.