USCO-Solver: Solving Undetermined Stochastic Combinatorial Optimization Problems

Guangmo Tong
Department of Computer and Information Sciences
University of Delaware
amotong@udel.edu

Abstract

Real-world decision-making systems are often subject to uncertainties that have to be resolved through observational data. Therefore, we are frequently confronted with combinatorial optimization problems of which the objective function is unknown and thus has to be debunked using empirical evidence. In contrast to the common practice that relies on a learning-and-optimization strategy, we consider the regression between combinatorial spaces, aiming to infer high-quality optimization solutions from samples of input-solution pairs – without the need to learn the objective function. Our main deliverable is a universal solver that is able to handle abstract undetermined stochastic combinatorial optimization problems. For learning foundations, we present learning-error analysis under the PAC-Bayesian framework using a new margin-based analysis. In empirical studies, we demonstrate our design using proof-of-concept experiments, and compare it with other methods that are potentially applicable. Overall, we obtain highly encouraging experimental results for several classic combinatorial problems on both synthetic and real-world datasets.

1 Introduction

Combinatorial optimization problems are not only of great theoretical interest but also central to enormous applications. Traditional research assumes that the problem settings (i.e., objective function) are completely known [1], but the reality is that the underlying system is often very complicated and only partial knowledge (from historical data) is provided. In a recommendation system, the item-user similarities could be unknown [2], which makes it impossible to compute the optimal recommendation scheme. In the study of wireless communications, the backbone network depends on stochastic node connections [3], incurring extra difficulties in designing management strategies. In the most general sense, we can formalize such scenarios by assuming that the objective function is governed by a *configuration space* associated with an *unknown* distribution, where our goal is to maximize the expected objective. For example, the configuration space may specify possible item-user similarities or candidate network structures. We call such problems as *undetermined stochastic combinatorial optimization (USCO) problems*.

Since the distribution of the configuration space is unknown, one can adopt the learning-and-optimization strategy [4], where the unknown distribution is first learned from data so that the subsequent optimization problem can be solved using existing methods. While such a strategy is natural and proved successful in several applications [5], it may require a large amount of data to learn an excessive number of parameters – for example, the weight between each pair of nodes in a network or the similarity between each pair of user and item in a recommendation system. In the worst case, we may not even have access to such kind of data (due to, for example, privacy issues

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

[6]). Furthermore, a more technical concern is that the learning process is often independent of the optimization process, causing the situation that optimizing the learned objective function does not produce a good approximation to the true objective function, which is theoretically possible [7]. These concerns motivate us to think of the settings that can eschew model learning and can address USCO problems aiming right at the approximation guarantees. To this end, we consider the regression between the input space and solution space, and wish to directly compute the solution for future inputs without learning the hidden objective function.

USCO-Solver. In this paper, we present USCO-Solver, a general-purpose solver to USCO problems. Our framework is designed through two main techniques: randomized function approximation and approximate structured prediction. The key idea is to construct a hypothesis space composed of affine combinations of combinatorial kernels generated using random configurations, from which a large-margin machine is trained using input-solution pairs via approximate inference. The main advantage of USCO-Solver is that it can handle an abstract USCO problem as long as an oracle to solve its deterministic counterpart is given. Another significance of USCO-Solver lies in our use of combinatorial kernels, which suggests a novel and principled way for incorporating an approximation algorithm into a learning process. In doing so, we are able to handle combinatorial objects easily without worrying much about their indifferentiability or the inherited symmetries, which is often not possible for mainstream learning methods [8].

Theoretical analysis. For USCO-Solver, we present a learning-error analysis under the PAC-Bayesian framework, where we introduce the multiplicative margin dedicated to bounding the approximation guarantees. In particular, we prove that the approximation ratio of the predictions is essentially bounded by $O(\alpha^2)$, whenever the considered problem admits an α-approximation in the deterministic sense. Such a result is possible due to the fact the hypothesis space of USCO-Solver can approximate the configuration space arbitrarily well. To our knowledge, this is the first result of such kind.

Empirical studies. We conduct experiments with USCO problems concerning three classic combinatorial objects: path, coverage, and matching. On various real-world datasets, we consistently observe that USCO-Solver not only works the way it is supposed to, but also outperforms other competitors by an evident margin in most cases. In many case studies, near-optimal solutions can be computed without consulting the true configuration distribution.

Supplementary material. The proofs, together with more results and discussions on the experiments, can be found in the supplementary material. In addition, we release the experimental materials, including source code, datasets, and pretrain models, as well as their instructions. The experiment materials can be found online[1].

2 Preliminaries

2.1 Undetermined stochastic combinatorial optimization problems

We are concerned with an abstract combinatorial optimization problem associated with three finite combinatorial spaces: the *input space* $\mathcal{X}$, the *output space* $\mathcal{Y}$, and a *configuration space* $\mathcal{C}$; in addition, we are given a bounded non-negative function $f(x, y, c) : \mathcal{X} \times \mathcal{Y} \times \mathcal{C} \to \mathbb{R}^+$ denoting the objective value to be maximized. Let Φ be the set of all distributions over $\mathcal{C}$. We consider *stochastic* combinatorial optimization problem in the sense that we seek to maximize f in terms of a distribution $\phi_{true} \in \Phi$ rather than a fixed configuration in $\mathcal{C}$. This is desired because the system that defines the objective values is often essentially probabilistic, which may stem from random networks or functions with random parameters. Therefore, the objective function is given by

$$F(x, y, \phi_{true}) = \int_{c \in \mathcal{C}} \phi_{true}(c) \cdot f(x, y, c)\, dc. \tag{1}$$

Given x and ϕ_{true}, we are interested in the problem

$$\max_{y \in \mathcal{Y}} F(x, y, \phi_{true}). \tag{2}$$

We may wish to compute either the optimal solution $H(x, \phi_{true}) \coloneqq \arg\max_{y \in \mathcal{Y}} F(x, y, \phi_{true})$ or its α-approximation $H_\alpha(x, \phi_{true})$ for some $\alpha \in (0, 1)$. Such problems are fundamental and also

[1] `https://github.com/cdslabamotong/USCO-Solver`

ubiquitous, ranging from the stochastic version of classic combinatorial problems [9] to applications subject to environmental uncertainties, such as commercial recommendation [10], viral marketing [11], and behavioral decision making in autonomous systems [12]. Taking the stochastic longest path problem [13] as an example, the length of each edge could follow a certain distribution, and therefore, each configuration $c \in \mathcal{C}$ corresponds to a weighted graph; in such a case, $x = (u, v)$ denotes a pair of source and destination, y denotes a path from u to v, and $f(x, y, c)$ amounts to the length of y in c.

While the above formulation is natural, we are always limited by our imperfect understanding of the underlying system, which could be formalized by assuming that the distribution ϕ_{true} over the configuration is not known to us. Since the true distribution ϕ_{true} is unknown, the objective function $F(x, y, \phi_{true})$ cannot be directly optimized. Essentially, a learning process is required. In such a sense, we call these problems *undetermined* stochastic combinatorial optimization (USCO) problems.

Oracle. In general, USCO problems are made difficult by a) the learning challenges in dealing with the unknown distribution ϕ_{true} and b) the approximation hardness in solving Equation 2. We focus on the learning challenges and thus assume the approximation hardness (if any) can be resolved by oracles. Notice that the objective function (Equation 1) lies in the positive affine closure of its discretizations, which is

$$\cup_{k=1}^{\infty} \Big\{ \sum_{i=1}^{k} w_i \cdot f(x, y, c_i) : w_i \in \mathbb{R}^+, c_i \in \mathcal{C} \Big\}. \tag{3}$$

For some fixed $\alpha \in (0, 1]$, we assume the access to a polynomial oracle for computing an α-approximation to maximizing any function in the above class, which implies that $H_\alpha(x, \phi)$ is obtainable for each $x \in \mathcal{X}$ and ϕ. We can think of α as the best ratio that one can obtain by a polynomial algorithm, with $\alpha = 1$ denoting the scenario when the optimal solution can be efficiently computed.

For a vector $\mathbf{w} = (w_1, ..., w_k) \in \mathbb{R}^k$ and a subset $C = \{c_1, ..., c_k\} \subseteq \mathcal{C}$ of configurations, we denote the induced kernel function as

$$\mathcal{K}_C(x, y) \coloneqq \Big(f(x, y, c_1), ..., f(x, y, c_k) \Big)$$

and the associated affine combination as

$$\hat{F}_{\mathbf{w},C}(x, y) \coloneqq \sum_{i=1}^{k} w_i \cdot f(x, y, c_i) = \mathbf{w}^\mathsf{T} \mathcal{K}_C(x, y).$$

Finally, we denote the α-approximation to $\max_{y \in \mathcal{Y}} \hat{F}_{\mathbf{w},C}(x, y)$ as $h^\alpha_{\mathbf{w},C}(x)$.

2.2 Learning settings

We consider samples of input-solution pairs:

$$S_m \coloneqq \Big\{ (x_i, y_i^\alpha) \Big\}_{i=1}^{m} \subseteq 2^{\mathcal{X} \times \mathcal{Y}}$$

where $y_i^\alpha \coloneqq H_\alpha(x_i, \phi_{true})$ is an α-approximation associated with the input x_i. Such samples intuitively offer the historical experience in which we successfully obtained good solutions to some inputs. From a learning perspective, we have formulated a regression task between two combinatorial spaces $\mathcal{X}$ and $\mathcal{Y}$. Some formal treatments are given as follows.

Let us assume that the true distribution ϕ_{true} is unknown but fixed, and there is a distribution $\mathcal{D}_\mathcal{X}$ over $\mathcal{X}$. Our goal is to design a learning framework $A_S : \mathcal{X} \to \mathcal{Y}$ that can leverage a set S of samples to compute a prediction $A_S(x) \in \mathcal{Y}$ for each future $x \in \mathcal{X}$. For a prediction $\hat{y} \in \mathcal{Y}$ associated with an input x, let $l(x, \hat{y}) \in [0, 1]$ be a general loss function. Since we aim to examine if $\hat{y}$ is a good approximation to Equation 2, the loss l is determined jointly by $\hat{y}$ and x. Consequently, we measure the performance of A by

$$\mathcal{L}(A_S, \mathcal{D}_\mathcal{X}, l) \coloneqq \mathbb{E}_{x \sim \mathcal{D}_X} \big[l(x, A_S(x)) \big]$$

3 A learning framework

In this section, we first present the learning framework and then analyze its generalization bounds. Finally, we discuss training algorithms.

3.1 USCO-Solver

Following the standard idea of structured prediction [14, 15, 16], we wish for a score function $\hat{F}(x, y)$ such that, for each $x \in \mathcal{X}$, the $y \in \mathcal{Y}$ that can maximize $\hat{F}(x, y)$ is a good solution to maximizing $F(x, y, \phi_{true})$. Suppose that we are provided with a set S_m of $m \in \mathbb{Z}$ samples. USCO-Solver consists of the following steps:

- **Step 1.** Select a distribution $\phi_{em} \in \Phi$ over $\mathcal{C}$ and decide a hyperparameter $K \in \mathbb{Z}$.
- **Step 2.** Sample K configurations $C_K = \{c_1, ..., c_K\} \subseteq \mathcal{C}$ independently following ϕ_{em}.
- **Step 3.** Compute a *seed vector* $\widetilde{\mathbf{w}} = (\widetilde{w}_1, ..., \widetilde{w}_K) \in \mathbb{R}^K$ using the training data.
- **Step 4.** Sample a vector $\overline{\mathbf{w}} \in \mathbb{R}^K$ from $Q(\mathbf{w}|\beta \cdot \widetilde{\mathbf{w}}, \mathcal{I})$, which is an isotropic Gaussian with identity convariance and a mean of $\beta \cdot \widetilde{\mathbf{w}}$, with β being

$$\beta \coloneqq \frac{4}{\min_p |\widetilde{w}_p| \cdot \alpha^2} \sqrt{2 \ln \frac{2mK}{\|\widetilde{\mathbf{w}}\|^2}}. \tag{4}$$

- **Score function.** With C_K and $\overline{\mathbf{w}} = (\overline{w}_1, ..., \overline{w}_K)$, we adopt the score function

$$\hat{F}_{\overline{\mathbf{w}}, C_K}(x, y) \coloneqq \sum_{i=1}^{K} \overline{w}_i \cdot f(x, y, c_i).$$

- **Inference.** For each input $x \in \mathcal{X}$, the prediction is given by $h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)$, which can be computed by the oracle.

Given that the framework is randomized, the error of interest is

$$\mathcal{L}(\text{USCO-Solver}_{S_m}, \mathcal{D}_{\mathcal{X}}, l) \coloneqq \mathbb{E}_{\overline{\mathbf{w}} \sim Q, C_K \sim \phi_{em}, x \sim \mathcal{D}_X} \left[l\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big) \right]. \tag{5}$$

So far, we have not seen how to determine the distribution ϕ_{em}, the parameter K, and the seed vector $\widetilde{\mathbf{w}}$. We will first analyze how they may affect the generalization bound (Sec. 3.2), and then discuss how to make selections towards minimizing the generalization bound (Sec. 3.3).

3.2 Generalization bound

We first study the general loss functions and then consider a particular loss function for measuring the approximation effect in terms of Equation 2.

3.2.1 General loss

The generalization error in general can be decomposed into training error and approximation error [17]; the PAC-Bayesian framework gives a particular bound of such kind [18, 19]. For USCO-Solver, since the decision boundary is probabilistic (as $\overline{\mathbf{w}}$ is sampled from the seed vector $\widetilde{\mathbf{w}}$), the training error has to be bounded by considering the amount by which $h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x_i)$ can deviate from $h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x_i)$ for each training input x_i. To this end, we measure the similarity of two outputs $y_1, y_2 \in \mathcal{Y}$ through the concept of margin, which is given by

$$\mathrm{m}(\mathbf{w}, C_K, x, y_1, y_2) \coloneqq \alpha \cdot \hat{F}_{\mathbf{w}, C_K}(x, y_1) - \hat{F}_{\mathbf{w}, C_K}(x, y_2). \tag{6}$$

For an input x, the potentially good solutions are those in

$$\mathrm{I}(x, \mathbf{w}, C_K) \coloneqq \left\{ y \in \mathcal{Y} : \mathrm{m}(\mathbf{w}, C_K, x, h^{\alpha}_{\mathbf{w}, C_K}(x), y) \leq \frac{\alpha}{2} \cdot \hat{F}_{\mathbf{w}, C_K}(x, h^{\alpha}_{\mathbf{w}, C_K}(x)) \right\}. \tag{7}$$

Intuitively, these are the solutions that are similar to the suboptimal one $h^{\alpha}_{\mathbf{w}, C_K}(x)$ in terms of the score function associated with $\mathbf{w}$ and C_K. For each $\mathbf{w}$, the training error is then taken by the worst-case loss over the possible y within the margin, which is

$$\mathcal{L}(\mathbf{w}, C_K, S_m) \coloneqq \frac{1}{m} \sum_{i=1}^{m} \max_{y \in \mathrm{I}(x_i, \mathbf{w}, C_K)} l(x_i, y).$$

Notice that the loss $\mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m)$ associated with the seed vector $\widetilde{\mathbf{w}}$ can be used to bound the training error with respect to $\overline{\mathbf{w}}$, provided that the predictions are within the margin (with high probability). With such intuitions, we have the first learning bound.

Theorem 1. *For each $\widetilde{\mathbf{w}}$, C_K, and $\delta > 0$, with probability at least $1-\delta$, we have*

$$\mathcal{L}(\textit{USCO-Solver}_{S_m}, \mathcal{D}_{\mathcal{X}}, l) \leq \mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m} + \sqrt{\frac{\ln \frac{2Km}{\|\widetilde{\mathbf{w}}\|^2}(\frac{4\|\widetilde{\mathbf{w}}\|}{\min_p |\widetilde{w}_p| \cdot \alpha^2})^2 + \ln \frac{m}{\delta}}{2(m-1)}}$$

The proof is inspired by the PAC-Bayesian framework [20, 21] applying to approximate structured prediction [22, 19], where our contribution lies in the analysis of the multiplicative margin.

3.2.2 Approximation loss

In line with Equation 2, one natural loss function is the approximation ratio of the predictions, which is

$$l_{\text{approx}}(x, \hat{y}) \coloneqq 1 - \min(\frac{F(x, \hat{y}, \phi_{true})}{F(x, H_\alpha(x, \phi_{true}), \phi_{true})}, 1) \in [0, 1]. \tag{8}$$

Essentially, we seek to generalize the approximation guarantee from the training samples to future inputs, and the guarantees on the training samples are limited by the oracle. With such, we hope to compare the generalization error under l_{approx} with α. According to Theorem 1, with an unlimited supply of training data (i.e., $m \to \infty$), the generalization error is bounded by $\mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m)$, and therefore, the learning error of the ERM solution is concentrated on

$$\inf_{\widetilde{\mathbf{w}}} \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{X}}} \Big[\max_{y \in \mathrm{I}(x_i, \widetilde{\mathbf{w}}, C_K)} l_{\text{approx}}(x_i, y) \Big]. \tag{9}$$

Relating the above quantity with α is not possible for the general case because the score function $\hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, y)$ defining the margin is independent of the objective function $F(x, y, \phi_{true})$ inducing the error. However, more concrete results can be obtained for the approximation loss, by leveraging a subtle relationship between our construction of the kernel function and the stochastic combinatorial optimization problem. The next result characterizes the relationship between Equation 9 and α.

Theorem 2. *Suppose that $f(x, y, c) \in [A, B]$ and $C \coloneqq \sup_c \frac{\phi_{true}(c)}{\phi_{em}(c)}$. For each $\epsilon > 0, \delta_1 > 0, \delta_2 > 0$ and ϕ_{em}, when K is no less than*

$$\frac{2 \cdot C^2 \cdot B^2}{\epsilon^2 \cdot \delta_2^2 \cdot A^2} \cdot \max(\frac{1}{2}, \ln |Y| + \ln \frac{1}{\delta_1}) \tag{10}$$

with probability at least $1-\delta_1$ over the selection of C_K, there exists a $\widetilde{\mathbf{w}}$ such that

$$\Pr_{x \sim \mathcal{D}_{\mathcal{X}}} \Big[\max_{y \in \mathrm{I}(x_i, \widetilde{\mathbf{w}}, C_K)} l_{\text{approx}}(x_i, y) \leq \frac{(1+\epsilon) - (1-\epsilon)(\alpha^2/2)}{(1+\epsilon)} \Big] \geq 1 - \delta_2.$$

The above result shows that the approximation ratio in generalization is essentially bounded by $\alpha^2/2$. It is intuitive that the bound on K depends on the deviation of ϕ_{em} from ϕ_{true} (i.e., C), the range of the objective function (i.e.. B/A), and the size of the output space.

3.3 Training algorithms

Theorems 1 and 2 have explained how m, K and ϕ_{em} may affect the generalization performance in theory. Practically, K can be conveniently taken as a hyperparameter, and ϕ_{em} is often the uniform distribution over $\mathcal{C}$ (unless prior knowledge about ϕ_{true} is given). Now, for the four steps in Sec. 3.1, the only problem left is to compute the seed vector $\widetilde{\mathbf{w}}$.

Relaxed margin. While Theorem 2 indicates that there exists a desired weight, we are not able to search for such a weight directly because the loss function l_{approx} is not accessible under our setting. In practice, one can adopt the zero-one loss or other loss function preferred by the considered problem. For example, when the output space is a family of sets, the loss function can be induced by the set similarity [23]. For a general loss function $l(x_i, y)$, Theorem 1 suggests computing the weight $\widetilde{\mathbf{w}}$ by minimizing the upper bound $\mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m}$:

$$\min_{\mathbf{w}} \max_{y} l(x_i, y) \cdot \mathbb{1}(y \in \mathrm{I}(x_i, \mathbf{w}, C_K)) + ||\mathbf{w}||^2,$$

where $\mathbb{1}$ is the indicator function. Notice that the above problem is not only non-convex but also complicated by the fact that $\mathbf{w}$ and $h^{\alpha}_{\mathbf{w},C_K}(x)$ are convoluted with each other. For ease of optimization, a relaxed margin will be useful, as seen shortly. Notice that we have $\hat{F}_{\mathbf{w},C_K}(x_i, h^{\alpha}_{\mathbf{w},C_K}(x_i)) \geq \alpha \cdot \hat{F}_{\mathbf{w},C_K}(x_i, y^{\alpha}_i)$, and therefore, for each $y \in \mathcal{Y}$, $y \in \mathrm{I}(x_i, \mathbf{w}, C_K)$ implies that

$$y \in \bar{\mathrm{I}}(x, \mathbf{w}, C_K) \coloneqq \big\{ y \in \mathcal{Y} : \frac{\alpha^2}{2} \cdot \hat{F}_{\mathbf{w},C_K}(x_i, y^{\alpha}_i) - \hat{F}_{\mathbf{w},C_K}(x_i, y) \leq 0 \big\},$$

where the new margin $\bar{\mathrm{I}}$ is measured with respect to y^{α}_i instead of $h^{\alpha}_{\mathbf{w},C_K}(x_i)$. Immediately, we have $\mathrm{I}(x_i, \mathbf{w}, C_K) \subseteq \bar{\mathrm{I}}(x_i, \mathbf{w}, C_K)$, which indicates that the error is further upper bounded by

$$\max_{y} l(x_i, y) \cdot \mathbb{1}(y \in \bar{\mathrm{I}}(x_i, \mathbf{w}, C_K)) + ||\mathbf{w}||^2, \tag{11}$$

which is less challenging to minimize.

Large-margin training. Seeking a large-margin solution and scaling the margin proportional to the loss function, Equation 11 amounts to solving the following problem:

$$\begin{aligned} \min \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2m} \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & \frac{\alpha^2}{2} \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y^{\alpha}_i) - \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y) \geq \eta \cdot l(x_i, y) - \xi_i, \ \forall i \in [m], \ \forall y \in \mathcal{Y}, y \neq y^{\alpha}_i \\ & \mathbf{w} \geq 0 \end{aligned}$$

where C and η are hyperparameters. Notably, this is the vanilla structured SVM [16, 24]. Viewing $\mathcal{K}_{C_K}$ as a score function, the constrains enforces that the solution y^{α}_i provided by the training data should have a high score. The above formulation appears to be a standard quadratic programming, but it in fact contains $|\mathcal{Y}|$ number of constrains, which can be prohibitively large (e.g., exponential in the coding length of the configuration). Since the constraints are equivalent to

$$\max_{y \in \mathcal{Y}} \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y) + \eta \cdot l(x_i, y) \leq \frac{\alpha^2}{2} \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y^{\alpha}_i) + \xi_i, \ \forall i \in [m],$$

the number of constraints could be reduced as linear in sample size if we can effectively solve the loss-augmented inference problem:

$$\max_{y \in \mathcal{Y}} \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y) + \eta \cdot l(x_i, y)$$

For the zero-one loss, this is exactly to solve $\max_{y \in \mathcal{Y}} \mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y)$, which, to our delight, can be solved using the oracle in Sec. 2.1. Once such loss-augmented inference problem can be solved, the entire programming can be readily solved by existing methods, such as the cutting-plane algorithm [25]. This completes the learning framework.

Remark 1. While our discussion so far is focused on maximization problems, the theoretical analysis can be adapted to minimization problems easily. In the training part for minimization problems, the major adaption is to reverse $\mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y^{\alpha}_i)$ and $\mathbf{w}^{\mathsf{T}} \mathcal{K}_{C_K}(x_i, y)$, which also turns the loss-augmented inference into a minimization problem.

Remark 2. For the stochastic shortest path problem, ϕ_{em} denotes a distribution over a collection $\mathcal{C}$ of weighted graphs, and assuming the weights are nonnegative, we may use Dijkstra algorithm as the oracle, as $\max_{y \in \mathcal{Y}} \hat{F}_{\mathbf{w},C}(x, y)$ amounts to computing the shortest path in the combining graph of the sampled weighted graphs.

4 Empirical studies

This section presents experiments across a variety of combinatorial problems, including stochastic shortest path (SSP), stochastic set cover (SSC), and stochastic bipartite matching (SBM). In addition to supporting the theoretical analysis, our empirical studies are of interest also because none of the existing methods is known to be effective for solving any USCO problem.

Table 1: **SSP results.** Each cell shows the mean of performance ratio with the standard deviation.

		UCSO-Solver					Other Methods		
Col	K	16	160	1600	3200	6400	NB	DSPN	Base
	ϕ_{exp}	1.942 (0.11)	1.952 (0.04)	1.565 (0.02)	1.129 (0.01)	1.148 (0.01)	2.064 (0.16)	2.009 (0.12)	1.956 (0.10)
	ϕ_{true}	1.300(0.02)	1.015 (0.01)	1.016 (0.01)	1.010 (0.01)	1.022 (0.01)			
NY	K	80	160	640	3200	6400	NB	DSPN	Base
	ϕ_{exp}	1.613 (0.01)	1.571 (0.02)	1.353 (0.01)	1.303 (0.02)	1.192 (0.01)	3.155 (0.03)	2.469 (0.07)	2.853 (0.28)
	ϕ_{true}	1.205 (0.01)	1.177 (0.01)	1.017 (0.01)	1.031 (0.01)	1.048 (0.01)			
Kro	K	160	1600	3200	6400	9600	NB	DSPN	Base
	ϕ_{exp}	7.599 (0.15)	5.829 (0.22)	5.858 (0.18)	4.613 (0.18)	4.323 (0.19)	10.08 (0.57)	7.451 (1.05)	10.92 (2.45)
	ϕ_{true}	1.361 (0.09)	1.026 (0.01)	1.022 (0.01)	1.042 (0.01)	1.051 (0.01)			

Evaluation metric. For each training pair (x_i, y_i^α) and a prediction $\hat{y}$, the performance ratio is defined as $\frac{F(x_i,y_i^\alpha,\phi_{true})}{F(x_i,\hat{y},\phi_{true})}$ for maximization problems and $\frac{F(x_i,\hat{y},\phi_{true})}{F(x_i,y_i^\alpha,\phi_{true})}$ for minimization problems. Therefore, lower is better in both cases. The performance is measured over five random testings. We here present the main experimental settings and discuss key observations, deferring the complete analysis to Appendix B.

4.1 Stochastic shortest path (SSP)

Problem definition and oracle. In the stochastic version of the shortest path problem, given a source u and a destination v in a graph $G = (V, E)$, we wish to find the path (from u to v) that is the shortest in terms of a distribution over the possible edge weights. In this case, the configuration space $\mathcal{C}$ denotes a family of weighted graphs associated with a distribution ϕ_{true}, the input $x = (u, v)$ is a pair of nodes, the output y is a path from u and v, and $f(x, y, c)$ denotes the length of y in graph c. We construct instances where the edge weights are non-negative, and thus, each function in Equation 3 can be minimized optimally using the Dijkstra algorithm [26], which is the oracle we use to generate samples and solve the inference problem in USCO-Solver.

Instance construction. To setup a concrete instance, we first fix a graph structure, where two real-world graphs (Col and NY) and one synthetic graph (Kro) are adopted. Col and NY are USA road networks complied from the benchmarks of DIMACS Implementation Challenge [27], and Kro is a Kronecker graph [28]. Following the common practice [29], we assign each edge a Weibull distribution with parameters randomly selected from $\{1, ..., 10\}$, which – together with the graph structure – induces the configuration space $\mathcal{C}$ and ϕ_{true}. For each instance, we generate the pool of all possible input-solution pairs (x, y), where, for an input $x = (u, v)$, y is the optimal solution to $\min_{y\in\mathcal{Y}} F(x, y, \phi_{true})$. For each considered method, we randomly select 160 training samples and 6400 testing samples from the pool.

Implementing USCO-Solver. We setup two realizations (ϕ_{exp} and ϕ_{true}) of ϕ_{em} that will be used in USCO-Solver. ϕ_{exp} assigns each edge in E an exponential distribution normalized in $[1, 1e5]$, and ϕ_{true} is exactly the true underlying distribution that defines the instance. Notice that ϕ_{true} is not obtainable under our learning setting, and it is used only to verify our designs. For each distribution, we sample a pool of 10000 configurations (i.e., weighted graphs). The number of configurations (i.e., K) is enumerated from small to large, with different ranges for different datasets. Given the size K, the configurations are randomly selected from the pool in each run of USCO-Solver. We use the zero-one loss, and the training algorithm is implemented based on Pystruct [30].

Implementing other competitors. For a baseline, we implement the Base method which, given an input $x = (u, v)$, outputs the shortest path from u to v in G where the edge weights are randomly sampled from $[0, 1]$. Conceptually, our problem can be perceived as a supervised learning problem from $V \times V$ to the space of the paths, and thus, off-the-shelf learning methods appear to be applicable, which however turns out to be non-trivial. We managed to implement two methods based respectively on Naive Bayes (NB) [31] and Deep Set Prediction Networks (DSPN) [32], where the idea is to

Table 2: **SSC results.** The table shows the results on two datasets Cora and Yahoo.

		UCSO-Solver					Other Methods		
Cora	K	8	16	160	320	640	GNN	DSPN	Rand
	ϕ_{uni}	1.480 (0.12)	1.452 (0.12)	1.230 (0.06)	1.147 (0.06)	1.083 (0.01)	1.036 (0.02)	1.782 (0.85)	16.57 (0.42)
	ϕ_{true}	1.029 (0.01)	1.033 (0.01)	1.003 (<0.01)	1.000 (<0.01)	1.000 (<0.01)			
Yahoo	K	8	16	160	320	640	GNN	DSPN	Rand
	ϕ_{uni}	1.294 (0.03)	1.356 (0.03)	1.151 (0.01	1.146 (0.07)	1.093 (0.03)	1.076 (0.10)	1.387 (0.142)	6.58 (0.17)
	ϕ_{true}	1.036 (0.02)	1.013 (0.01)	1.003 (<0.01)	1.009 (<0.01)	0.999 (<0.01)			

leverage them to learn the node importance in the shortest path and then build the prediction based on the node importance.

Observations. The main results are presented in Table 1. We highlight two important observations: a) the performance of USCO-Solver smoothly increases when more configurations are provided, and b) when fed with configurations from the true distribution ϕ_{true}, USCO-Solver can easily output near-optimal solutions on all the datasets. These observations confirm that USCO-Solver indeed functions the way it is supposed to. In addition, the efficacy of USCO-Solver is significant and robust in terms of the performance ratio. On datasets like Col and NY, it can produce near-optimal solutions using ϕ_{exp}. Meanwhile, other methods are not comparable to USCO-Solver, though DSPN offers non-trivial improvement compared with Base on NY and Kro. It is also worth noting that different instances exhibit different levels of hardness with respect to the number of configurations needed by USCO-Solver. 3200 configurations are sufficient for USCO-Solver to achieve a low ratio on Col, while the ratio is still larger than 4.0 on Kro even though 9600 configurations have been used. See Appendix B.2 for a complete discussion.

4.2 Stochastic set cover (SSC)

In network science, coverage problems often require to compute a set of terminals that can maximally cover the target area [33]; in applications like recommendation system or document summarization [34], one would like to select a certain number of items that can achieve the maximum diversity by covering topics as many as possible. These problems can be universally formulated as a maximal coverage problem [35], where each instance is given by a family U of subsets of a ground set, where our goal is to select a k-subset of U of which the union is maximized. In its stochastic version, we assume that an item will appear in a subset with a certain probability [36]. The oracle we use is the greedy algorithm, which is a $(1 - 1/e)$-approximation [37]. We construct two instances based on two real-world bipartite graphs Cora [38] and Yahoo [39]. Similar to the setup for SSP, we first generate the ground truth distribution ϕ_{true} and then generate samples. We implement two learning methods based on Graph neural network (GNN) and DSPN, together with a baseline method Rand that randomly selects the nodes in $\hat{y}$. For USCO-Solver, we consider a distribution ϕ_{uni} that generates a configuration by generating subsets uniformly at random. See Appendix B.3 for details.

The results are presented in Table 2. First, the observations here are similar to those in the SSP problem – USCO-Solver works in a manner as we expect, and it can produce high-quality solutions when a sufficient number of configurations are provided. We can see that GNN and DSPN are also effective compared with Rand, which suggests that, compared to SSP, SSC is much easier for being handled by existing learning methods.

4.3 Stochastic bipartite matching (SBM)

Our last problem is bipartite matching, which is regularly seen in applications such as public housing allocation [40], semantic web service [41] and image feature analysis [42]. We consider the minimum weight bipartite matching problem. Given a weighted bipartite graph $G = (L, R, E)$, the input $x = (L^*, R^*)$ consists of two subsets $L^* \subseteq L$ and $R^* \subseteq R$ with $|L^*| = |R^*|$, and the output y is a perfect matching between L^* and R^* such that the total cost is minimized. In other words, y is a bijection between L^* and R^*. In its stochastic version, the weight w_e for each edge $e \in E$ follows a certain distribution, and we aim to compute the matching that can minimize the expected

Table 3: **SBM results with ϕ_{uni} and ϕ_{true}.**

K	16	160	1600	3200	6400	12800	19200	Rand
ϕ_{uni}	3.627 (0.01)	3.582 (0.01)	3.407 (0.01)	3.261 (0.01)	3.107 (0.01)	2.874 (0.01)	2.696 (0.01)	3.670 (0.01)
ϕ_{true}	1.029 (0.01)	1.033 (0.01)	1.003 (<0.01)	1.000 (<0.01)	1.000 (<0.01)			

Table 4: **SBM results with ϕ_{10}, ϕ_5, ϕ_1 and $\phi_{0.3}$.**

	16	160	320	640		16	160	320	640
ϕ_{10}	4.889 (0.09)	3.602 (0.06)	1.831 (0.04)	1.271 (0.01)	ϕ_5	4.321 (0.07)	1.403 (0.01)	1.120 (0.01)	1.040 (<0.01)
ϕ_1	1.038 (<0.01)	1.008 (<0.01)	1.003 (<0.01)	1.002(<0.01)	$\phi_{0.3}$	1.008 (<0.01)	1.003 (<0.01)	1.002 (<0.01)	1.002(<0.01)

cost $F(x, y, \phi_{true})$. Notice that the minimum weight bipartite matching problem can be solved in polynomial time by linear programming [43], which is a natural oracle to use in USCO-Solver. We adopt a synthetic bipartite graph with 128 nodes. The weight w_e for each edge $e \in E$ follows a Gaussian distribution $\mathcal{N}(\mu_e, \sigma_e)$ with μ_e sampled uniformly from $[1, 10]$ and $\sigma_e = 0.3 \cdot \mu_e$, which induces the true distribution ϕ_{true}. See Appendix B.4 for details.

Results under ϕ_{uni}. For USCO-Solver, we consider ϕ_{uni} that generates the configuration c by giving each edge a weight from $[1, 10]$ uniformly at random. We also test USCO-Solver with features from the true distribution ϕ_{true}. The baseline method Rand produces y for a given input $x = (L^*, R^*)$ by randomly generating a permutation of R^*. The result of this part is shown in Table 3. We see that USCO-Solver can again produce a better ratio when more configurations are provided, but different from SSP and SSC, it cannot achieve a ratio that is close to 1 even when 19200 configurations have been used. Plausibly, this is because the output space of SBM is more structured compared with SSP and SSC.

Incorporating prior knowledge into USCO-Solver. For problems like bipartite matching where uniform configurations cannot produce a near-optimal solution, it would be interesting to investigate if USCO-Solver can easily benefit from domain expertise. To this end, given a parameter $q \in \{0.3, 1, 5, 10\}$, we consider the distribution ϕ_q that samples the weight for edge e from $[\mu_e - q \cdot \mu_e, \mu_e + q \cdot \mu_e]$, which accounts for the case that a confidence interval of the weight w_e is known. The performance ratios produced by such distributions are shown in Table 4. As we can see from the table, the result indeed coincides with the fact that the prior knowledge is strong when p is small. More importantly, with the help of such prior knowledge, the ratio can be reduced to nearly 1 using no more than 160 configurations under ϕ_1, while under the uniform distribution ϕ_{uni}, the ratio was not much better than Rand with the same amount of configurations (as seen in Table 3).

5 Further discussions

Related work. The learning-and-optimization framework has been used widely (e.g., [44, 45, 46, 47, 48]). For example, Du *et al.* [49] and He *et al.* [50] study the problem of learning influence function which is used in influence maximization; recently, Wilder *et al.* [51] proposes a decision-focused learning framework based on continuous relaxation of the discrete problem. In contrast to these works, we consider the input-solution regression without attempting to learn the objective function. A similar setting was adopted in [52] for studying a special application in social network analysis, but our paper targets abstract problems and provides generalization bounds on the approximation ratio. There have been recent efforts to develop learning methods that can handle combinatorial spaces (e.g., [53, 54, 32]), where the key is to preserve the inherited combinatorial structures during the learning process. Our research is different because the problem we consider involves a hidden optimization problem; in other words, we target the optimization effect rather than the prediction accuracy. Our research is also related to the existing works that attempt to solve combinatorial optimization problems using reinforcement learning [55, 56, 57, 58]; however, the objective function is known in their settings, while we study the model-free setting.

Limitations. Our theoretical analysis is based on the assumption that y is an approximation solution in each input-solution (x, y), but in practice, it may be impossible to prove such guarantees for real datasets. Thus, it is left to experimentally examine USCO-Solver using the training pairs (x, y) where

y is produced using heuristic but not approximation algorithms. In another issue, we have shown that USCO-Solver is effective for three classic combinatorial problems, but it remains unknown if USCO-Solver is universally effective, although it is conceptually applicable to an arbitrary USCO problem. In addition, our experiments do not rule out the possibility that off-the-shelf learning methods can be effective for some USCO problems (e.g., SSC), which suggests a future research direction. Finally, we hope to extend our research by studying more practical applications involving highly structured objects, such as Steiner trees, clustering patterns, and network flows. See Appendix C for a more comprehensive discussion.

Acknowledgments and Disclosure of Funding

We thank the reviewers for their time and insightful comments. Funding in direct support of this work: support from the University of Delaware.

References

[1] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[2] H.-N. Kim and A. El Saddik, "A stochastic approach to group recommendations in social media systems," *Information Systems*, vol. 50, pp. 76–93, 2015.

[3] Y. Deng, L. Wang, M. Elkashlan, A. Nallanathan, and R. K. Mallik, "Physical layer security in three-tier wireless sensor networks: A stochastic geometry approach," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1128–1138, 2016.

[4] E. Horvitz and T. Mitchell, "From data to knowledge to action: A global enabler for the 21st century," *arXiv preprint arXiv:2008.00045*, 2020.

[5] E. Horvitz, "From data to predictions and decisions: Enabling evidence-based healthcare," *Computing Community Consortium*, vol. 6, 2010.

[6] A. Askinadze and S. Conrad, "Respecting data privacy in educational data mining: an approach to the transparent handling of student data and dealing with the resulting missing value problem," in *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2018, pp. 160–164.

[7] E. Balkanski, A. Rubinstein, and Y. Singer, "The limitations of optimization from samples," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 1016–1027.

[8] Y. Zhang, J. Hare, and A. Prügel-Bennett, "Fspool: Learning set representations with featurewise sort pooling," in *International Conference on Learning Representations*, 2019.

[9] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, no. 2, pp. 239–287, 2009.

[10] Y. Kim and K. Shim, "Twitobi: A recommendation system for twitter using probabilistic modeling," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011, pp. 340–349.

[11] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, pp. 5–es, 2007.

[12] W. Chen and L. Kloul, "Stochastic modelling of autonomous vehicles driving scenarios using pepa," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2019, pp. 317–331.

[13] E. Ando, H. Ono, K. Sadakane, and M. Yamashita, "Computing the exact distribution function of the stochastic longest path length in a dag," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2009, pp. 98–107.

[14] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine learning*, vol. 75, no. 3, pp. 297–325, 2009.

[15] D. Weiss and B. Taskar, "Structured prediction cascades," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 916–923.

[16] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 896–903.

[17] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[18] D. McAllester, "Generalization bounds and consistency," *Predicting structured data*, pp. 247–261, 2007.

[19] Y. Wu, M. Lan, S. Sun, Q. Zhang, and X. Huang, "A learning error analysis for structured prediction with approximate inference," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6131–6141.

[20] D. A. McAllester, "Some pac-bayesian theorems," *Machine Learning*, vol. 37, no. 3, pp. 355–363, 1999.

[21] D. McAllester, "Simplified pac-bayesian margin bounds," in *Learning theory and Kernel machines*. Springer, 2003, pp. 203–215.

[22] A. Kulesza and F. Pereira, "Structured learning with approximate inference," 2008.

[23] S. Zhou, J. Wang, R. Shi, Q. Hou, Y. Gong, and N. Zheng, "Large margin learning in set-to-set similarity comparison for person reidentification," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 593–604, 2017.

[24] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer, "Large margin methods for structured and interdependent output variables." *Journal of machine learning research*, vol. 6, no. 9, 2005.

[25] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Machine learning*, vol. 77, no. 1, pp. 27–59, 2009.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[27] C. Demetrescu, A. V. Goldberg, and D. S. Johnson, "Implementation challenge for shortest paths." *Encyclopedia of Algorithms*, vol. 15, p. 54, 2008.

[28] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: an approach to modeling networks." *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

[29] H. Rinne, *The Weibull distribution: a handbook*. CRC press, 2008.

[30] A. C. Müller and S. Behnke, "Pystruct: learning structured prediction in python." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2055–2060, 2014.

[31] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[32] Y. Zhang, J. Hare, and A. Prugel-Bennett, "Deep set prediction networks," in *NeurIPS*, 2019, pp. 3207–3217.

[33] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *Mobile networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[34] H. Lin and J. Bilmes, "A class of submodular functions for document summarization," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 510–520.

[35] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[36] A. Deshpande, L. Hellerstein, and D. Kletenik, "Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover," in *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2014, pp. 1453–1466.

[37] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005.

[38] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[39] "Yahoo! search marketing advertiser bidding data." https://webscope.sandbox.yahoo.com.

[40] N. Benabbou, M. Chakraborty, X.-V. Ho, J. Sliwinski, and Y. Zick, "Diversity constraints in public housing allocation," in *17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, 2018.

[41] U. Bellur and R. Kulkarni, "Improved matchmaking algorithm for semantic web services based on bipartite graph matching," in *IEEE international conference on web services (ICWS 2007)*. IEEE, 2007, pp. 86–93.

[42] Y.-Q. Cheng, V. Wu, R. Collins, A. R. Hanson, and E. M. Riseman, "Maximum-weight bipartite matching technique and its application in image feature matching," in *Visual Communications and Image Processing'96*, vol. 2727. International Society for Optics and Photonics, 1996, pp. 453–462.

[43] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[44] A. N. Elmachtoub and P. Grigas, "Smart "predict, then optimize"," *Management Science*, 2021.

[45] Y.-h. Kao, B. Roy, and X. Yan, "Directed regression," *Advances in Neural Information Processing Systems*, vol. 22, pp. 889–897, 2009.

[46] P. L. Donti, B. Amos, and J. Z. Kolter, "Task-based end-to-end model learning in stochastic optimization," *arXiv preprint arXiv:1703.04529*, 2017.

[47] G.-Y. Ban and C. Rudin, "The big data newsvendor: Practical insights from machine learning," *Operations Research*, vol. 67, no. 1, pp. 90–108, 2019.

[48] D. Bertsimas and N. Kallus, "From predictive to prescriptive analytics," *Management Science*, vol. 66, no. 3, pp. 1025–1044, 2020.

[49] N. Du, Y. Liang, M. Balcan, and L. Song, "Influence function learning in information diffusion networks," in *International Conference on Machine Learning*. PMLR, 2014, pp. 2016–2024.

[50] X. He, K. Xu, D. Kempe, and Y. Liu, "Learning influence functions from incomplete observations," *arXiv preprint arXiv:1611.02305*, 2016.

[51] B. Wilder, B. Dilkina, and M. Tambe, "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1658–1665.

[52] G. Tong, "Stratlearner: Learning a strategy for misinformation prevention in social networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[53] H. Maron, O. Litany, G. Chechik, and E. Fetaya, "On learning sets of symmetric elements," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6734–6744.

[54] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NIPS*, 2017.

[55] E. B. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *NIPS*, 2017.

[56] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[57] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, "Reinforcement learning for solving the vehicle routing problem," *arXiv preprint arXiv:1802.04240*, 2018.

[58] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, 2020.

[59] C. McDiarmid, "On the method of bounded differences," *Surveys in combinatorics*, vol. 141, no. 1, pp. 148–188, 1989.

[60] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[61] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[62] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, pp. 820–830, 2018.

[63] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3744–3753.

[64] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3523–3532.

[65] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen, "3d steerable cnns: Learning rotationally equivariant features in volumetric data," *arXiv preprint arXiv:1807.02547*, 2018.

[66] A. Parmentier, "Learning to approximate industrial problems by operations research classic problems," *Operations Research*, 2021.

[67] M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolínek, "Differentiation of blackbox combinatorial solvers," *arXiv preprint arXiv:1912.02175*, 2019.

[68] Y. Bengio, E. Frejinger, A. Lodi, R. Patel, and S. 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*. Springer, 2020, pp. 99–111.

[69] V. Nair, D. Dvijotham, I. Dunning, and O. Vinyals, "Learning fast optimizers for contextual stochastic integer programs." in *UAI*, 2018, pp. 591–600.

[70] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 237–252.

[71] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2722–2730.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section ?.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Sec. 5
   (c) Did you discuss any potential negative societal impacts of your work? [No] We study abstract stochastic combinatorial problems, focusing on learning foundations. Thus, we believe our research is foundational and not tied to particular applications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] They are in appendix.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Due to the 100MB limitation, we provide code, data, and pretrain models for selected experiments in the supplemental material. The entire project will be published later after the reviewing period.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] To our best, we have provided all the training details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report standard deviations.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Such information can be found in appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We use public dataset and source code following their licenses.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Our dataset is purely numerical and does not have any personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing, and our research is not related to any human subject.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

USCO-Solver: Solving Undetermined Stochastic Combinatorial Optimization Problems (Appendix)

A Proofs

A.1 Proof of Theorem 1

To prove Theorem 1, the following result (proved in Sec. A.1.1) would be useful.

Lemma 1. *For each $\widetilde{\mathbf{w}}$, C_K and ϕ_{true}, with probability at least $1 - \frac{\|\widetilde{\mathbf{w}}\|^2}{m}$ over the selection of $\overline{\mathbf{w}}$, we have*

$$l(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)) \leq \max_{y \in \mathrm{I}(x, \widetilde{\mathbf{w}}, C_K)} l(x, y) \tag{12}$$

holding simultaneously for all $x \in X$.

Lemma 1 implies that for each x_i we have

$$\begin{aligned}\mathbb{E}_{\overline{\mathbf{w}} \sim Q}\left[l(x_i, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x))\right] &\leq (1 - \frac{\|\widetilde{\mathbf{w}}\|^2}{m}) \max_{y \in \mathrm{I}(x_i, \widetilde{\mathbf{w}}, C_K)} l(x_i, y) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m} \max_y l(x_i, y) \\ &\leq \max_{y \in \mathrm{I}(x_i, \widetilde{\mathbf{w}}, C_K)} l(x_i, y) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m}. \end{aligned} \tag{13}$$

Notice that to prove Theorem 1, it suffices to show that for each C_K, we have

$$\mathbb{E}_{\overline{\mathbf{w}} \sim Q(\mathbf{w}|\widetilde{\mathbf{w}}), x \sim \mathcal{D}_X}\left[l(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x))\right] \leq \mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m} + \sqrt{\frac{\frac{\beta^2 \|\widetilde{\mathbf{w}}\|^2}{2} + \ln \frac{m}{\delta}}{2(m-1)}}, \tag{14}$$

from which we can obtain Theorem 1 by plugging in Equation 4. To prove Equation 14, notice that the PAC-Bayesian approach [18, 20] immediately yields the following results for our settings:

Theorem 3. *For each C_K, let $\mathcal{P}$ be a prior distribution over the hypothesis space (i.e., $\mathbb{R}^K$) and let $\delta \in (0, 1)$. Then, with probability of at least $1 - \delta$ over the choice of an iid training set $S = \{z_1, ..., z_m\}$ sampled according to D, for all distributions $\mathcal{Q}$ over $\mathbb{R}^K$, we have*

$$\mathbb{E}_{\overline{\mathbf{w}} \sim \mathcal{Q}, x \sim \mathcal{D}_X}\left[l(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x))\right] \leq \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{\overline{\mathbf{w}} \sim \mathcal{Q}}\left[l(x_i, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x_i))\right] + \sqrt{\frac{D(\mathcal{Q} \,||\, \mathcal{P}) + \ln(m/\delta)}{2(m-1)}}$$

where $D(\mathcal{Q} \,||\, \mathcal{P})$ is the KL divergence.

Taking $\mathcal{Q}$ as the distribution Q we defined in Sec. 3.1, by Equation 13, we have

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{\overline{\mathbf{w}} \sim \mathcal{Q}}\left[l(x_i, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x_i))\right] \leq \mathcal{L}(\widetilde{\mathbf{w}}, C_K, S_m) + \frac{\|\widetilde{\mathbf{w}}\|^2}{m}.$$

Taking $\mathcal{P}$ as isotropic Gaussian with identify covariance and a mean of zero, $D(\mathcal{Q} \,||\, \mathcal{P})$ is analytically given by $\frac{\beta^2 \|\widetilde{\mathbf{w}}\|^2}{2}$, which completes the proof.

A.1.1 Proof of Lemma 1

Since $h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x) \in \mathrm{I}(x, \widetilde{\mathbf{w}}, C_K)$ would imply Equation 12, it suffices to show that

$$\Pr_{\overline{\mathbf{w}} \sim Q}\left[h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x) \in \mathrm{I}(x, \widetilde{\mathbf{w}}, C_K)\right] \geq 1 - \frac{\|\widetilde{\mathbf{w}}\|^2}{m}. \tag{15}$$

For a vector $\mathbf{w}$, let us denote by w_p its p-th entry. By the definition of $Q(\mathbf{w}|\beta \cdot \widetilde{\mathbf{w}}, \mathcal{I})$, for each $p \in \{1, ..., K\}$ and $\epsilon > 0$, we have

$$\Pr\left[|\overline{w}_p - \beta \cdot \widetilde{w}_p| \geq \epsilon \cdot \beta \cdot \widetilde{w}_p\right] \leq 2 \exp(\frac{-(\beta \cdot \widetilde{w}_p \cdot \epsilon)^2}{2}). \tag{16}$$

Setting $\epsilon = \frac{\alpha^2}{4}$, retrieving β through Equation 4, and taking the union bound for p over $\{1, ..., K\}$, we have $|\overline{w}_p - \beta \cdot \widetilde{w}_p| \leq \epsilon \cdot \beta \cdot \widetilde{w}_p$ holding simultaneously for all p with probability at least $1 - \frac{\|\widetilde{\mathbf{w}}\|^2}{m}$. Now, to prove Lemma 1, it suffices to show that

$$|\overline{w}_p - \beta \cdot \widetilde{w}_p| \leq \epsilon \cdot \beta \cdot \widetilde{w}_p, \forall p \in \{1, ..., K\} \implies h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x) \in \mathrm{I}(x, \widetilde{\mathbf{w}}, C_K). \tag{17}$$

With the LHS of Equation 17, we have

$$\begin{aligned}
&\mathrm{m}(\widetilde{\mathbf{w}}, C_K, x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x), h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)) \\
&\{\text{by Equation 6}\} \\
=&\frac{1}{\beta} \cdot \beta \cdot \Big(\alpha \cdot \hat{F}_{\widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) - \hat{F}_{\widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the linearity of } \hat{F}_{\mathbf{w}, C_K} \text{ with respect to } \mathbf{w}\} \\
=&\frac{1}{\beta} \cdot \Big(\alpha \cdot \hat{F}_{\beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) - \hat{F}_{\beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the linearity of } \hat{F}_{\mathbf{w}, C_K} \text{ with respect to } \mathbf{w}\} \\
=&\frac{1}{\beta} \cdot \Big(\alpha \cdot \hat{F}_{\beta \cdot \widetilde{\mathbf{w}} - \overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) + \alpha \cdot \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) \\
&\qquad - \hat{F}_{\beta \cdot \widetilde{\mathbf{w}} - \overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big) - \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the LHS of Equation 17}\} \\
\leq&\frac{1}{\beta} \cdot \Big(\alpha \cdot \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) + \alpha \cdot \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) \\
&\qquad + \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big) - \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the definition of } h^{\alpha}_{\mathbf{w}, C}(x)\} \\
\leq&\frac{1}{\beta} \cdot \Big(\alpha \cdot \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) + \alpha \cdot \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) \\
&\qquad + \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x)\big) - \alpha \cdot \hat{F}_{\overline{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the definition of } h^{\alpha}_{\mathbf{w}, C}(x)\} \\
\leq&\frac{1}{\beta} \cdot \Big(\alpha \cdot \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big) + \frac{1}{\alpha} \cdot \hat{F}_{\epsilon \cdot \beta \cdot \widetilde{\mathbf{w}}, C_K}\big(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)\big)\Big) \\
&\{\text{by the linearity of } \hat{F}_{\mathbf{w}, C_K} \text{ with respect to } \mathbf{w}\} \\
=&(\epsilon \cdot \alpha) \cdot \hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)) + \frac{\epsilon}{\alpha} \cdot \hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)) \\
&\{\text{since } \alpha \leq 1\} \\
\leq&\frac{2\epsilon}{\alpha} \cdot \hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)) \\
&\{\text{by the setting of } \epsilon\} \\
\leq&\frac{\alpha}{2} \cdot \hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, h^{\alpha}_{\widetilde{\mathbf{w}}, C_K}(x)).
\end{aligned}$$

The above inequality implies $h^{\alpha}_{\overline{\mathbf{w}}, C_K}(x) \in \mathrm{I}(x, \widetilde{\mathbf{w}}, C_K)$ due to the definition (i.e., Equation 7).

A.2 Proof of Theorem 2

The generalization of the approximation ratio relies on a guaranteed function approximation, as shown in the next lemma.

Lemma 2. *For each $\epsilon, \delta_1, \delta_2 \in \mathbb{R}$ and ϕ_{em}, when K satisfies Equation 10, with probability at least $1 - \delta_1$ over the selection of C_K, there exist a $\widetilde{\mathbf{w}}$ such that*

$$\Pr_{x \sim \mathcal{D}_{\mathcal{X}}}\Big[|\hat{F}_{\widetilde{\mathbf{w}}, C_K}(x, y) - F(x, y, \phi_{true})| \leq \epsilon \cdot F(x, y, \phi_{true}),\ \forall y \in \mathcal{Y}\Big] \geq 1 - \delta_2. \tag{18}$$

For each $x \in \mathcal{X}$ and $\widetilde{\mathbf{w}}$, suppose that $|\hat{F}_{\widetilde{\mathbf{w}},C_K}(x,y) - F(x,y,\phi_{true})| \leq \epsilon \cdot F(x,y,\phi_{true})$ is true for each $y \in \mathcal{Y}$. Then for each $y \in \mathrm{I}(x,\widetilde{\mathbf{w}},C_K)$, we have

$$\begin{aligned}
&\mathrm{m}(\widetilde{\mathbf{w}},C_K,x,h^{\alpha}_{\widetilde{\mathbf{w}},C_K}(x),y,\alpha) \leq \frac{\alpha}{2}\cdot \hat{F}_{\widetilde{\mathbf{w}}}(x,h^{\alpha}_{\widetilde{\mathbf{w}},C_K}(x)) \\
\implies &\hat{F}_{\widetilde{\mathbf{w}},C_K}(x_i,y) \geq \frac{\alpha}{2}\cdot \hat{F}_{\widetilde{\mathbf{w}},C_K}(x_i,h^{\alpha}_{\widetilde{\mathbf{w}},C_K}(x_i)) \\
\implies &\hat{F}_{\widetilde{\mathbf{w}},C_K}(x_i,y) \geq \frac{\alpha^2}{2}\cdot \hat{F}_{\widetilde{\mathbf{w}},C_K}(x_i,H(x_i,\phi_{true})) \\
\implies &(1+\epsilon)\cdot F(x_i,y,\phi_{true}) \geq \frac{\alpha^2}{2}\cdot(1-\epsilon)\cdot F(x,H(x_i,\phi_{true}),\phi_{true})
\end{aligned}$$

implying

$$l_{\approx}(x_i,y) = 1 - \frac{F(x,y,\phi_{true})}{F(x,H(x,\phi_{true}),\phi_{true})} \leq \frac{(1+\epsilon) - \frac{\alpha^2}{2}\cdot(1-\epsilon)}{(1+\epsilon)}.$$

Now we are left to prove Lemma 2.

A.2.1 Proof of Lemma 2

For convenience, the McDiarmid's Inequality is repeated, as follows.

Definition 1 (McDiarmid's Inequality [59])**.** Let X_1,..., X_m be independent random variables with domain $\mathcal{X}$. Let $f : \mathcal{X}^m \to \mathbb{R}$ be a function that satisfies

$$|f(x_1,...,x_i,...,x_m) - f(x_1,...,x_i^{'},...,x_m)| \leq a_i$$

for each i and $x_1,...,x_m,x_i^{'} \in \mathcal{X}$. The for each $\epsilon > 0$, we have $\Pr[f - \mathbb{E}[f] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum a_i^2}\right)$.

Recall that ϕ_{em} is the distribution generating $C_K = \{c_1,...,c_K\}$, and ϕ_{true} is the truth distribution. Notice that c_i are iid variables, and for each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, we have the key observation that

$$\mathbb{E}_{c\sim\phi_{em}}\left[\frac{\phi_{true}(c)}{\phi_{em}(c)}f(x,y,c)\right] = F(x,y,\phi_{true}) \tag{19}$$

For each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, let us denote the average function as

$$h_{(x,y)}(c_1,...,c_K) = \sum_{i=1}^{K}\frac{\phi_{true}(c_i)}{K\cdot\phi_{em}(c_i)}f(x,y,c_i).$$

and the normalized average function as

$$\hat{h}_{(x,y)}(c_1,...,c_K) = \frac{1}{F(x,y,\phi_{true})}\sum_{i=1}^{K}\frac{\phi_{true}(c_i)}{K\cdot\phi_{em}(c_i)}f(x,y,c_i).$$

For each y, we will analyze the concentration of $\hat{h}$ on its mean value with respect to $\mathcal{D}_{\mathcal{X}}$. To this end, let us consider the function

$$g_y(c_1,...,c_K) = \sqrt{\int_{x\in\mathcal{X}}\left(\hat{h}_{(x,y)}(c_1,...,c_K) - 1\right)^2 d\,\mathcal{D}_{\mathcal{X}}(x)}. \tag{20}$$

The stability of $g_y(c_1,...,c_K)$ can be established as follow. For each $c_1,...,c_K,c_* \in \mathcal{C}$ and $i \in [K]$, replacing c_i by c_i^*, the change is bounded by

$$|g_y(c_1,...,c_i,...,c_K) - g_y(c_1,...,c_i*,...,c_K)| \tag{21}$$

$$\leq\sqrt{\int_{x\in\mathcal{X}}\left(\hat{h}_{(x,y)}(c_1,...,c_i,...,c_K) - \hat{h}_{(x,y)}(c_1,...,c_i^*,...,c_K)\right)^2 d\,\mathcal{D}_{\mathcal{X}}(x)} \leq \frac{2\cdot C\cdot B}{K\cdot A} \tag{22}$$

For the expectation of $g_y(c_1, ..., c_K)$ with respect to c_i, we have due to the Jensen's inequality that

$$\mathbb{E}_{C_K \sim \phi_{em}}[g_y(c_1, ..., c_K)] \leq \sqrt{\mathbb{E}_{C_K \sim \phi_{em}}[g_y(c_1, ..., c_K)^2]}$$
$$= \sqrt{\int_{x \in \mathcal{X}} \mathbb{E}_{C_K \sim \phi_{em}}[\Big(\hat{h}_{(x,y)}(c_1, ..., c_K) - 1\Big)^2] d\,\mathcal{D}_{\mathcal{X}}(x)}$$

Because $\mathbb{E}_{C_K \sim \phi_{em}}[\Big(\hat{h}_{(x,y)}(c_1, ..., c_K) - 1\Big)^2]$ is the variance of an average of iid random variables, it can be derived as $\frac{1}{K}(\mathbb{E}_{c \sim \phi_{em}}[\Big(\frac{1}{F(x,y,\phi_{true})} \frac{\phi_{true}(c)}{\phi_{em}(c)} f(x, y, c)\Big)^2] - 1)$, which is bounded by $\frac{C^2 \cdot B^2}{A^2 \cdot K}$. Therefore, we have

$$\mathbb{E}_{C_K \sim \phi_{em}}\Big[g_y(c_1, ..., c_K)\Big] \leq \frac{C \cdot B}{A \cdot \sqrt{K}}. \tag{23}$$

Let $\epsilon_0 = \frac{\epsilon \cdot \delta_2}{2}$. Applying the McDiarmid's Inequality, we have

$$\Pr_{C_K \sim \phi_{em}}\Big[g_y(c_1, ..., c_K) - \epsilon_0 \geq \epsilon_0\Big] \tag{24}$$
$$\leq \Pr_{C_K \sim \phi_{em}}\Big[g_y(c_1, ..., c_K) - \frac{C \cdot B}{A \cdot \sqrt{K}} \geq \epsilon_0\Big] \tag{25}$$
$$\leq \Pr_{C_K \sim \phi_{em}}\Big[g_y(c_1, ..., c_K) - \mathbb{E}[g_y(c_1, ..., c_K)] \geq \epsilon_0\Big] \tag{26}$$
$$\leq \exp\big(\frac{-K \cdot \epsilon_0^2 \cdot A^2}{2 \cdot C^2 \cdot B^2}\big) \leq \delta_1/|Y|. \tag{27}$$

Taking the union bound over $y \in \mathcal{Y}$, the above result implies that with probability at least $1 - \delta_1$, we have for all $y \in \mathcal{Y}$ that

$$\sqrt{\int_{x \in \mathcal{X}} \Big(\hat{h}_{(x,y)}(c_1, ..., c_K) - 1\Big)^2 d\,\mathcal{D}_{\mathcal{X}}(x)} \leq 2 \cdot \epsilon_0 \tag{28}$$

Consequently, by Jensen's inequality, we have

$$\mathbb{E}_{x \sim \mathcal{D}_{\mathcal{X}}}\Big[|\hat{h}_{(x,y)}(c_1, ..., c_K) - 1|\Big] \leq \sqrt{\mathbb{E}_{x \sim \mathcal{D}_{\mathcal{X}}}[|\hat{h}_{(x,y)}(c_1, ..., c_K) - 1|^2]} \leq 2 \cdot \epsilon_0 \tag{29}$$

simultaneously for all $y \in \mathcal{Y}$. Using Markov's inequality, this implies

$$\Pr_{x \sim \mathcal{D}_{\mathcal{X}}}\Big[|\frac{h_{(x,y)}(c_1, ..., c_K)}{F(x, y, \phi_{true})} - 1| \leq \epsilon,\ \forall y \in \mathcal{Y}\Big] \leq \frac{2\epsilon_0}{\epsilon} = \delta_2,$$

meaning that $\widetilde{\mathbf{w}} = (\frac{\phi_{true}(c_1)}{K \cdot \phi_{em}(c_1)}, ..., \frac{\phi_{true}(c_K)}{K \cdot \phi_{em}(c_K)})$ is the desired weight.

B Empirical Evaluation

This section presents additional information about experimental settings and discusses observations complementary to the main paper.

B.1 General Information

Computing platform. Our experiments are executed on Amazon EC2 C5 Instance with 96 virtual CPUs and memory 192G. The training and testing in all the experiments are reasonably fast.

Implementation. USCO-Solver is implemented based on Pystruct [30]; GNN and DSPN are implemented based on original source code. DSPN is proposed in [32] where the main modules are input encoder, set encoder, and set decoder. Following [52], we encode the input as a set of elements where the feature of each element is the associated one-hot vector. The input encoder and set encoder are MLP with three hidden layers of size 512. The inner optimization is performed for ten steps with

a rate of $1e6$ in each round, and the outer loop is optimized with Adam with a learning rate of 0.01. For GNN, we adopt the model in the seminal work [60], with two graph convolution layers followed by an MLP of size (512, 512, 256) with ReLU as the activation function. GNN and DSPN are trained with early stopping according to the validation data. For USCO-Solver, we report the results under the weights that give the best prime objective value in the one-slack cutting plane algorithm [25].

Data. The dataset information is shown in Table 5, where Bipartite is the graph we used in the SBM problem.

Table 5: **Datasets.**

	Kro	NY	Col	Cora	Yahoo	Bipartite
Nodes	1024	768	512	3785	10152	128
Edges	2655	1590	1118	5429	52567	16384
Total input-solution pairs	570954	137196	395266	10000	10000	10000
Default training size	160	160	160	80	80	160
Default testing size	6400	6400	6400	640	640	640
Graph structure source	[28]	[27]	[27]	[38]	[39]	Synthetic

Open-Source, Dataset, and Reproducibility. The instances, input-solution pairs, pretrain models, and source code are available in the supplementary files. We release the pretrain model by giving the configurations (e.g., weighted graphs) and the learned weights. However, the supplementary material is limited to 100MB, while our entire dataset has been more than 150GB, which is mainly because the configurations are space-consuming. Thus, at the submission time, we only include the configuration pool of Cora for the SSP problem, and will release the entire project later on GitHub.

Training complexity. We adopt the standard one-slack cutting-plane algorithm (Algorithm 3 in [25]) for training, which involves a hyperparameter ϵ (in addition to the hyperparameters C and η in our formulation) to control the stopping criteria. Following directly from Corollary 2 in [25], for fixed C, η and ϵ, the time complexity is $O(m * A + m^2 + mK + K^3)$ where m is the number of training instances, A is the running time of the oracle (in Sec 2.1), and K is the number of sampled configurations. We adopt the implementation provided by [30], where we have $C = 0.0001$, $\eta = 1$, and $\epsilon = 0.001$. We believe that training complexity is an important issue, and we will add such discussions to the updated version.

B.2 Stochastic shortest path

Oracle. Given G and ϕ_{true}, for each $x = (u, v)$, solving $\arg\min_y F(x, y, \phi_{true})$ amounts to finding the shortest path in G where each edge e is weighted as $\mathbb{E}[w_e]$ with respect to ϕ_{true}, which is due to the linearity of expectation. Therefore, this can be solved by the Dijkstra algorithm. For each $C_K = \{c_1, ..., c_K\}$ and a weight $\mathbf{w}$, minimizing $\sum_{i=1}^{K} w_i \cdot f(x, y, c_i)$ is equivalent to finding a shortest path in G where each edge e is weighted as $\sum_i c_i(e)$, with $c_i(e)$ being the weight of e in configuration c_i. Therefore, it can be solved again by the Dijkstra algorithm.

Implementing NB and DSPN. To make NB and DSPN applicable, we first take the output path as a set of nodes and view our problem as a supervised learning problem from $V \times V$ to 2^V. NB solves this problem by learning $\Pr[x|y] \Pr[y]$. In addition to the Naive Bayes assumption, we assume the nodes in y are also independent, and therefore, we are to learn $\prod_{u \in x} \prod_{v \in y} \Pr[u|v] \Pr[u]$. In this way, for each x, NB finds a sequence of nodes ordered by their likelihoods. Similar to NB, DSPN produces a sequence of nodes ordered by the values in the output layer. After getting the ordered sequence for each input $x = (u, v)$, we add the nodes one by one to the graph following the given order until u and v are connected, and then report the path connecting u to v. That is, we consider the path with the nodes that are the most probable to appear in the shortest path.

Additional experimental results. We have also tested USCO-Solver with more training samples, and the results are given in Table 6. Overall, we see that the performance does not increase very much when more training samples are available. In addition, excessive training samples can incur extra difficulties in solving the quadratic programming, and thus can sometimes hurt the performance,

which is evidenced by the results on NY. We also test USCO-Solver with distribution ϕ_{norm}, which generates the edge weights using Gaussian $\mathcal{N}(0,1)$. Compared to ϕ_{exp}, the configurations from ϕ_{norm} lack diversity because the edges are likely to have the same weight, which can again hurt the performance. The results of ϕ_{norm} are also given in Table 6, and we can observe that more configurations are needed by USCO-Solver to achieve a decent performance ratio.

Table 6: **Additional SSP results.**

			USCO-Solver					
		Train-Size	16	160	1600	3200	6400	
	ϕ_{exp}	160	1.942 (0.11)	1.952 (0.04)	1.565 (0.02)	1.129 (0.01)	1.148 (0.01)	
Col	ϕ_{exp}	320	1.893 (0.11)	1.676 (0.02)	1.123 (0.03)	1.097 (0.01)	1.152 (0.02)	
	ϕ_{exp}	1600	1.882 (0.05)	1.710 (0.06)	1.382 (0.05)	1.106 (0.01)	1.095 (0.01)	
		Train-Size	80	160	640	3200	6400	
	ϕ_{exp}	160	1.613 (0.01)	1.571 (0.02)	1.353 (0.01)	1.303 (0.02)	1.192 (0.01)	
NY	ϕ_{exp}	1600	1.989 (0.03)	1.723 (0.03)	1.552 (0.03)	1.315 (0.01)	1.194 (0.01)	
		Train-Size	80	160	640	3200	6400	9600
	ϕ_{norm}	160	13.29 (0.35)	11.29 (0.34)	7.625 (0.25)	6.030 (0.07)	4.765(0.02)	4.436 (0.02)

B.3 Stochastic set cover

Problem definition. The maximum coverage problem can be modeled as follows. Consider a bipartite graph $G = (L, R, E)$ where L and R denote the node sets and E denotes the edges. Given a set of nodes $R^* \subseteq R$ and a budget $k \in \mathbb{Z}$, we wish to solve

$$\underset{y \subseteq L, |y| \leq k}{\arg\max} \; | \cup_{v \in y} N(v) \cap x|,$$

where $N(v) \subseteq R$ denotes the neighbors of v. In its stochastic version, we assume that each edge $e \in E$ will appear in the graph with probability p_e. In line with the notations in Sec. 2, the configuration space $\mathcal{C}$ now denotes a family of bipartite graphs associated with a distribution ϕ_{true} induced by p_e, the input $x = (R^*, k)$ consists of a subset $R^* \subseteq R$ and the budget k, the output $y \subseteq L$ is a set of nodes with $|y| \leq k$, and, $f(x, y, c) = | \cup_{v \in y} N_c(v) \cap x|$ denotes the coverage, where $N_c(v) \subseteq R$ is the neighbor set of v in c.

Oracle Notice that $f(x, y, c)$ is submodular with respect to y for each c and x. Therefore, $F(x, y, \phi_{true})$ and $\sum_{i=1}^{k} w_i \cdot f(x, y, c_i)$ are also submodular, as submodularity is persevered under non-negative summation, making the greedy algorithm a natural oracle [37].

Instance construction. We adopt two graphs Cora and Yahoo. Given the graph structure, for each edge e, we generate the ground truth p_e by sampling two integers a and b from $[1, ..., 10]$ and assigning p_e as $a/(a + b)$. For each instance, to generate one input-solution pair $(x = (R^*, k), y)$, we first determine $|R^*|$ by sampling an integer from a power-law distribution with a parameter 2.5 and a scale 200 [61] and set the budget k as $\lfloor 0.1 * |R^*| \rfloor$; after getting the input, we compute y by approximating the true objective function F using the greedy algorithm.

Implementing USCO-Solver and other competitors. For USCO-Solver, we consider ϕ_{uni} that generates configurations by keeping each edge in the graph with a probability of 0.1. In addition, we also test USCO-Solver with features from the true distribution ϕ_{true}. By treating the considered problem as a supervised learning problem from 2^R to 2^L, we also implement two learning methods based on GNN and DSPN. We encode the nodes as one-hot vectors and select the top k nodes according to the output layer.

About GNN seed value. On Cora, the GNN results in Table 3 are computed based on five runs with seed values $\{17, 19, 23, 44, 13\}$. On Yahoo, the results are based on five runs with seed values $\{7, 13, 37, 42, 53\}$.

B.4 Stochastic bipartite matching

Instance construction. We adopt a complete bipartite graph G with 128 nodes. The weight w_e for each edge $e \in E$ follows a Gaussian distribution $\mathcal{N}(\mu_e, \sigma_e)$ with μ_e sampled uniformly from $[1, 10]$ and $\sigma_e = 0.3 \cdot \mu$, which induces the true distribution ϕ_{true}. To generate one input-solution pair $(x = (L^*, R^*), y)$, we first determine $|L^*|$ by sampling an integer from the power-law distribution with a parameter of 2.5 and a scale of 200 [61], and then select the nodes for L^* and R^* randomly from G; after getting the input x, we compute y using linear programming [61].

Additional experimental results. Table 7 shows the results when more configurations and larger training sets are used. The results show that the performance indeed increases with training size, but the improvement is significant only if the configurations are sufficiently many. Intuitively, more training samples are useful only if the models are sufficiently representative.

Table 7: **Additional SBM results.**

	USCO-Solver				
	Train-Size	6400	12800	19200	25600
ϕ_{uni}	160	3.107 (0.04)	2.874 (0.02)	2.696 (0.02)	2.544 (0.02)
ϕ_{uni}	320	2.991 (0.02)	2.692 (0.03)	2.559 (0.02)	2.378 (0.02)
ϕ_{uni}	640	2.966 (0.02)	2.661 (0.03)	2.407 (0.02)	2.278 (0.02)

C Related work and future work

The learning-and-optimization framework has been used widely (e.g., [44, 45, 46, 47, 48]). For example, Du *et al.* [49] and He *et al.* [50] study the problem of learning influence function which is used in influence maximization; recently, Wilder *et al.* [51] proposes a decision-focused learning framework based on continuous relaxation of the discrete problem. In contrast to these works, we consider the input-solution regression without attempting to learn the objective function. A similar setting was adopted in [52] for studying a special application in social network analysis, while our paper targets abstract problems and provides generalization bounds on the approximation ratio.

There have been recent efforts to develop learning methods that can handle combinatorial spaces, where the key is to preserve the inherited combinatorial structures during the learning process. One major direction aims to design universal approximators through neural networks (e.g., [53, 54]), and another promising method is to leverage the invariance of gradients [32]. Our research is different because the problem we consider involves a hidden optimization problem; in other words, we target the optimization effect rather than the prediction accuracy. While the USCO-Solver presented in this paper does not utilize any deep architecture, the performance might be further improved by using structure-preserving representation learning methods (e.g., [62, 63, 64, 65]), which we leave for future work.

Our research is also related to the existing works that attempt to solve combinatorial optimization problems using reinforcement learning [55, 56, 57, 58]; however, the objective function is known in their settings, while we study the model-free setting. For another future direction, it is possible that USCO-Solver can benefit from the existing methods by solving the loss-augmented inference problem through reinforcement learning, which is often more efficient than approximation algorithms.

Additional related works. Parmentier [66] presents an interesting framework that solves hard combinatorial problems by aligning the solutions of the target problem with the solutions to an easy problem; in contrast, our paper seeks the alignment between inputs and their optimal solutions. Vlastelica *et al.* [67] focuses on computing the gradient of the learning framework composed of neural networks and combinatorial processes. This is indeed a promising method for allowing USCO-Solver to handle extra features associated with the inputs or solutions. In our paper, we only consider the input and solutions that are pure combinatorial objects. Bengio *et al.* [68] studies the problem of approximating the instance of two-stage stochastic integer programming using representative scenarios, which is conceptually similar to [66] in that they both attempt to connect one optimization problem with another one. Different from our paper, a) the training phase in [68] requires to know the original instance, and b) [68] focuses on approximating the instance parameters (analogous to

the true distributions under our context) rather than the optimization effect. Nair *et al.* [69] designs a reinforcement learning method to solve the two-stage stochastic integer programming, where the objective function is given approximately by iid samples.

Limitations. Our theoretical analysis is based on the assumption that y is an approximation solution in each input-solution pair (x, y), but in practice, it may be impossible to prove such guarantees for real datasets. Thus, it is left to experimentally examine USCO-Solver using the training pairs (x, y) where y is produced using heuristic but not approximation algorithms. In another issue, we have shown that USCO-Solver is effective for three classic combinatorial problems, but it remains unknown if USCO-Solver is universally effective, even though it is conceptually applicable to an arbitrary USCO problem. It should also be noted that off-the-shelf learning methods may work very well for some USCO problems (e.g., the SSC problem), and therefore, it is worth investigating the situation where deep neural networks are effective for solving USCO problem. Finally, we hope to extend our research by studying more practical applications involving highly structured objects, such as Steiner tree, clustering pattern, and network flow. For example, direct perception in autonomous systems [70, 71] falls into our settings, in the sense that we need to infer a mapping from the results of the perception system to driving decisions – the input and output space are both combinatorial and highly structured.