Benchmarking PtO and PnO Methods in the Predictive Combinatorial Optimization Regime

Haoyu Geng[1†], **Hang Ruan**[1†], **Runzhong Wang**[2], **Yang Li**[1], **Yang Wang**[3], **Lei Chen**[3], **Junchi Yan**[1*]

[1]Dept. of CSE & School of AI & Moe Key Lab of AI, Shanghai Jiao Tong University
[2]Massachusetts Institute of Technology
[3]Finvolution Group
{genghaoyu98, zzrh01, yanglily, yanjunchi}@sjtu.edu.cn
runzhong@mit.edu, {wangyang09, chenlei04}@xinye.com

Abstract

Predictive combinatorial optimization, where the parameters of combinatorial optimization (CO) are unknown at the decision-making time, is the precise modeling of many real-world applications, including energy cost-aware scheduling and budget allocation on advertising. Tackling such a problem usually involves a prediction model and a CO solver. These two modules are integrated into the predictive CO pipeline following two design principles: "Predict-then-Optimize (PtO)", which learns predictions by supervised training and subsequently solves CO using predicted coefficients, while the other, named "Predict-and-Optimize (PnO)", directly optimizes towards the ultimate decision quality and claims to yield better decisions than traditional PtO approaches. However, there lacks a systematic benchmark of both approaches, including the specific design choices at the module level, as well as an evaluation dataset that covers representative real-world scenarios. To this end, we develop a modular framework to benchmark 11 existing PtO/PnO methods on 8 problems, including a new industrial dataset for combinatorial advertising that will be released. Our study shows that PnO approaches are better than PtO on 7 out of 8 benchmarks, but there is no silver bullet found for the specific design choices of PnO. A comprehensive categorization of current approaches and integration of typical scenarios are provided under a unified benchmark. Therefore, this paper could serve as a comprehensive benchmark for future PnO approach development and also offer fast prototyping for application-focused development. The code is available at `https://github.com/Thinklab-SJTU/PredictiveCO-Benchmark`.

1 Introduction

Predictive combinatorial optimization is a family of Combinatorial Optimization (CO) problems where the problem parameters are unknown during decision-making. Predictive CO models a wide range of real-world settings, including energy cost-aware scheduling [67], budget allocation for website information dissemination [72], portfolio optimization [47]. In predictive CO, the optimization itself could be solved by readily available solvers (e.g. Gurobi [28]) once the problem parameters are predicted. However, due to the inherent uncertainty and noise from the real world, the prediction is never perfect, resulting in potentially misleading problem parameters. Even the optimal solution under these parameters could be a bad decision in the real world. As shown in Fig 1(a~b), job scheduling without considering energy price fluctuations may result in energy waste and higher costs.

---

*Correspondence author. † denotes equal contribution. This work was in part supported by NSFC 92370201, 72342023.

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

(a) Energy cost-aware scheduling   **(b) Energy price visualization**   **(c) PtO vs PnO**

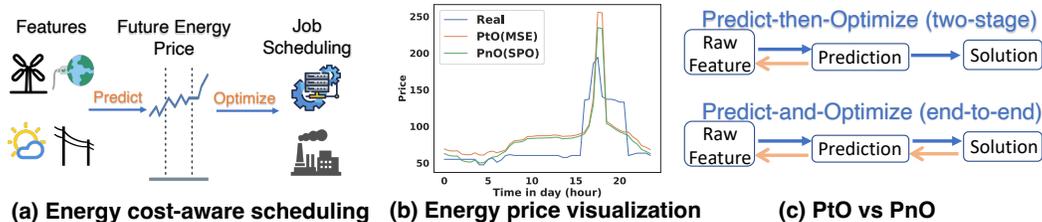



Figure 1: (a) Example of predictive CO in energy-cost aware scheduling. Factories deploy job schedules based on energy price predictions to reduce production costs. (b) Visualization of energy prices of the 200th test instance in SEMO dataset. PtO makes improper predictions, which further prescribes sub-optimal decisions. (c) PtO vs. PnO. PnO designs decision-oriented training approaches that emerged recently as a promising direction to tackle predictive CO.

Hence, there are important and practical needs to jointly consider both prediction and decision-making for predictive optimization (especially CO) problems [20, 46, 9]. A basic solution for predictive CO is "**predict-then-optimize**" [7] (**PtO**, or called the two-stage approach), which first predicts coefficients of the optimization task through a predictive model trained under the supervision of ground truth coefficients and subsequently utilizes off-the-shelf solvers to obtain solutions. Intuitively, it is expected that a higher prediction accuracy would result in better decision quality. Nonetheless, as also evidenced by Figure 5(c~d), there often exists a gap between prediction objectives and ultimate decision goals, leading to suboptimal decisions by PtO models. Therefore, as shown in Fig 1(b), a recent series of studies [46, 68] have proposed the new "**predict-and-optimize**" paradigm (**PnO**, or decision-focused learning [72, 44, 60] that learns to directly cater for the ultimate decision objectives. In cases where the quality of the final decision is paramount, joint prediction-and-optimization approaches may be more direct and beneficial. Though this can sometimes come at the expense of prediction accuracy (as is shown in Fig 5(c)), they exhibit notable improvement over PtO on several optimization tasks.

The main challenges towards predict-and-optimize for predictive CO lie in two aspects: (1) the solution's derivatives with respect to the optimization coefficients are not available. (2) the variables in the optimization problems are discrete. Both challenges lead to the blocking of gradients. To mitigate this issue, several approaches have been proposed, including designing proxy loss functions [46], gradient approximation [54], etc. However, there lacks a systematic categorization of existing methods, and it is not clear which methods are effective for specific problems and scenarios. Moreover, the experimental benchmarks are relatively fragmented, and the existing proposed models have not undergone comprehensive evaluation, impeding the community's progression. Though some implementations [1, 64] are available for some methods, they only support linear or convex problems, while overlooking other non-linear and submodular problems. Moreover, the optimization datasets are limited in scale, and some of them use generated data for the prediction part, lacking an industrially large-scale real-world dataset for validating the PnO performance.

In this work, we systematically review existing methodologies and establish a benchmark to align each model. Based on empirical results, we then conduct a series of in-depth experiments to explore key factors for PnO model design, and find several useful tips for deployment of PnO models. Based on these findings, we provide a set of recommendations for practitioners on model selection when considering the applicability and practical performance. We anticipate that our open-source benchmark and new dataset will gain increased attention, thereby fostering advancements in both research and practical applications. Our contributions and conclusions are summarized as follows:

- We systematically review current PnO approaches and categorize them into four categories according to how the problem is solved regarding the decision variables: discrete or continuous; and how the loss function is designed: (statistically) direct or surrogate. We also give a PnO model choice recommendation by analyzing the potential gains and challenges during the real-world deployment.

- We develop a modular framework of 11 existing PtO/PnO methods on 8 problems, and multiple solvers for a fair benchmarking. We also include a new industrial dataset regarding the combinatorial advertising problem, formulated as an integer linear program with uncertain conversion rates.

- Our benchmark results demonstrate PnO is better than PtO on 7 out of 8 predictive CO problems. However, **no silver bullet is found for specific design choices**, suggesting the necessity of trial-and-

2

Table 1: Categorization of representative methods, where "Predictive" belongs to "PtO" while others are "PnO"s. BB, ID, CPLayer is short for Blackbox, Identity, and cvxpy layer. "✘" means it supports the characteristic in some models among many. Suppose N is the decision variable size, and K is the number of extra optimization samples. T_{LODL} denotes backward time following [60].

	Predictive	Discrete			Continuous			Statistical	Surrogate	
On-the-fly	✓		✓			✓		✗	✗	✘
$\partial \mathbf{z}/\partial \mathbf{y}$	None	Innate	Interpolation		Automatic differentiation		Designed	Statistical	Learned	
Approach	**Two-stage**	**DFL**	**BB**	**ID**	**QPTL**	**CPLayer**	**SPO**	**NCE, LTR**	**LODL**	**SurCO**
Cite	[7]	[60]	[54]	[58]	[72]	[1]	[46]	[49],[44]	[60]	[23]
Objective	Any	Any	Linear	Linear	Quadratic	Convex	Linear	Any	Any	Nonlinear
Constraints	Any	Any	Linear	Linear	Linear	Convex	Linear	Any	Any	Integer
Requires $y^\star$	✓	✓	✗	✗	✗	✗	✓	✓	✓	✘
Additional solving	0	0	1	0	0	0	1	K	K	$K \cdot |\mathcal{D}|$
Backward Complexity	$O(1)$	$O(1)$	$O(N)$	$O(1)$	$O(N^3)$	$O(N^3)$	$O(N)$	$O(KN)$	$O(K \cdot T_{LODL})$	$O(N^2)$

error for various scenarios. Therefore, our comprehensive and modular-based benchmark covering mainstream PnOs could further help quick prototyping in application-focused development.

● Our extensive benchmark discovers some key factors for PnO methodology design, suggesting future research directions. Specifically, we try to answer 3 research questions regarding **relationship between prediction accuracy and decision quality**, **impacts of prediction labels on PnO**, and **versatility of PnO across different settings**. The experiments indicate that leveraging decision information for more favorable trade-offs may be a key factor in how PnO works; pre-training with predicted labels can enhance certain PnO methods; the versatility of PnO in optimization parameters still needs improvement.

2 Problem Formulation

Consider a predictive combinatorial optimization with unknown coefficients $\mathbf{y}$:

$$\max_{\mathbf{z} \in \mathcal{Z}} \quad f(\mathbf{z}, \mathbf{y}, \boldsymbol{\theta}) \; s.t. \; \mathbf{z} \in \text{Constr}(\boldsymbol{\theta}) \,, \tag{1}$$

where $f(\mathbf{z}, \mathbf{y}, \boldsymbol{\theta})$ denotes the known and closed-formed optimization objective (abbreviated as $f(\mathbf{z}, \mathbf{y})$ below) with discrete variable $\mathbf{z} \in \mathcal{Z}$, and $\mathbf{y}$ is collection of unknown optimization coefficients, $\boldsymbol{\theta}$ are optimization parameters that are known and fixed, and decision variables $\mathbf{z}$ obey the constraints $\text{Constr}(\boldsymbol{\theta})$. We assume that the coefficients of the constraints are known in line with the majority of literature [72, 60, 44]. We denote the solution obtained through one solver call for a problem instance with coefficient $\mathbf{y}$ as $\mathbf{z}(\mathbf{y})$, and $\mathbf{z}^\star(\mathbf{y})$ as the optimal solution. The solvers can be commercial solvers (e.g. Gurobi [28]), open-sourced solvers (e.g. cvxpy [16]), or neural solvers (e.g. submodular [37]).

Though coefficients $\mathbf{y}$ are unknown, in many circumstances, they could be estimated by a predictive model using a collection of historical or pre-collected dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, where $\mathbf{x}$ denotes relevant raw features. Therefore, the learning objective of the prediction step is:

$$\min_{\mathcal{M}} \mathbb{E}_{(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{D}}[\mathcal{L}_{pred}(\hat{\mathbf{y}}_i, \mathbf{y}_i)] \,, \tag{2}$$

where $\mathcal{L}_{pred}$ is a training loss specified by the prediction output, e.g. mean squared error (MSE). Suppose the prediction model $\mathcal{M}$ serves as a mapping from the feature vector $\mathbf{x}_i$ to coefficients $\hat{\mathbf{y}}_i$, i.e. $\hat{\mathbf{y}}_i = \mathcal{M}(\mathbf{x}_i)$, the predictive optimization problem [72] is:

$$\max_{\mathbf{z}_i \in \mathcal{Z}} \quad \mathbb{E}_{(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{D}} [f(\mathbf{z}_i(\mathcal{M}(\mathbf{x}_i)), \mathcal{M}(\mathbf{x}_i), \boldsymbol{\theta})] \,. \tag{3}$$

The evaluation of the PtO could be critical. In many circumstances where the real coefficients of the test set are available, regret [46, 75, 26, 44] is used to evaluate decision quality. Let the decision quality [60, 23] of solution $\hat{\mathbf{z}}$ be the objective under the ground-truth coefficient $\mathbf{y}$:

$$\text{DQ}(\hat{\mathbf{z}}) = f(\hat{\mathbf{z}}, \mathbf{y}, \boldsymbol{\theta}). \tag{4}$$

Then the regret could be obtained by the difference of the decision quality with solutions under the estimated coefficient($\mathbf{z}^\star(\mathbf{y})$,) and ground-truth coefficients ($\mathbf{z}^\star(\hat{\mathbf{y}})$):

$$\text{Regret}(\hat{\mathbf{y}}, \mathbf{y}) = |f(\mathbf{z}^\star(\mathbf{y}), \mathbf{y}, \boldsymbol{\theta}) - f(\mathbf{z}^\star(\hat{\mathbf{y}}), \mathbf{y}, \boldsymbol{\theta})|. \tag{5}$$

3

Problems
Linear: Knapsack · Energy Scheduling · Bipartite Matching · Combinatorial Advertising
Non-linear: TopK · Budget Allocation · Portfolio

Predictors: Linear regression · Multi-Layer Perceptron · ResNet

Solvers: Gurobi · Cvxpy · Heuristic · Neural

Losses
Discrete: DFL, BB, ID
Continuous: Cvxpy layer SPO+
Statistical: NCE, LTR
Surrogate: LODL

Evaluations
Metrics: Regret · Uplift
Configs: Var Size · Constraint

Figure 3: A modular code framework supporting 11 problems, 8 PtO/PnO models, multiple solvers, and various evaluations under configurable parameters. Users can easily customize their own problems, predictors, models, and solvers.

However, in many cases (e.g. combinatorial advertising problem), the ground truth coefficients **y** are not readily available or even impossible to obtain, so the evaluation of PtO becomes challenging. Online A/B testing may be one approach; however, it is associated with higher costs and risks in practice. Consequently, we propose using uplift, an offline controlled variable metric common in causal inference tasks, as an alternative evaluation elaborated in Appendix C.7.3.

3 Categorization of Existing Methods

As briefly introduced earlier, the major challenges of PnO lie in two aspects: (1) unavailable/non-informative gradients and (2) discrete variables. These two challenges both block the gradient backpropagation so that the prediction model cannot be updated by the final objective. Table 1 categorizes existing works into four groups by how the optimization is solved and how the gradient is obtained, where the gradient flow of these approaches is shown in Figure 2. We consider the **PtO** training as a baseline method while concurrently noting that the first challenge can be readily addressed if the solver is differentiable, such as neural solvers for top-k [73] or Sinkhorn algorithm [62] used in neural graph matching [70]. The technical details are given in Appendix B.3.

3.1 The Discrete Category

As discussed in [54, 6, 50], the gradient of decision variable **z** with respect to coefficients **y** revealed as piecewise constant functions (almost zero everywhere, and undefined otherwise). To address this issue, the "discrete" category solves the optimization with the original discrete solver in the forward pass, while in the backward pass, the gradients are estimated via designed interpolation functions, such as linear interpolation done by Blackbox [54]. Representative models include **DFL** [60], **Blackbox** [54], **Perturb** [6] and **I-MLE** [50], **Identity** [58], etc.

One advantage of this category is that the gradient interpolations work on the fly and do not require coefficient labels or additional optimization samples. Therefore, this class of methods requires the least additional resources. However, they are only adaptable to linear or convex optimization objectives, since gradients of more complex objectives are hard to estimate. Though a vanilla DFL method is agnostic to the optimization form, it is prone to have high variances [60] and our experiments also indicate such issues (shown in Figure 6). Besides, some methods require additional solver calls in the forward pass, like BlackBox, Perturb, and I-MLE, which adds a bit of computational overhead.

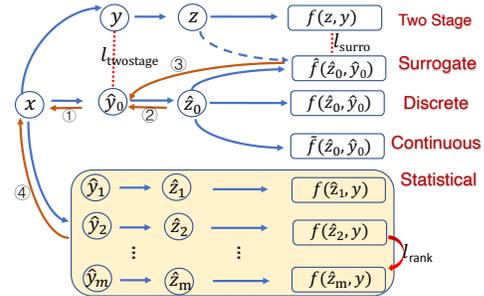



Figure 2: Gradient flow of existing PnO methods, where f, $\hat{f}$ and $\tilde{f}$ denote the original objective, learned objective by surrogate model, and the continuous relaxation of the original objective, respectively. The path of back propagation of the vanilla two-stage is ①, while discrete/continuous categories use ①②, statistical one uses ④, and surrogate one uses ①③. l_{surro} represents the surrogate losses to measure how the surrogate imitates the original optimization objective, while l_{rank} refers to the designed loss that encodes solution ranking—e.g., ensuring that the optimal solution is assigned a lower loss than suboptimal ones.

4

3.2 The Continuous Category

The "continuous" category estimates the gradient of the relaxed version of the CO problem since continuous problems are implicitly differentiable in nature. In the forward pass, it generally solves the relaxed optimization problem, whereas in the backward pass, the gradients are obtained by automatic differentiation (of such as KKT conditions). Representative models include **OptNet** [3], **CPLayer** [1] and more [18, 40, 19] for continuous PnOs and **QPTL** [72], **IntOpt** [45], SPO-relax [46] (short as SPO below), and more [68, 22, 52], for PnO on discrete problems.

The advantage of this category is that the gradient $\partial z/\partial y$ is readily available when the problem is relaxed to continuous. However, it necessitates tailored relaxations for each individual problem, and not every discrete problem has readily available or straightforward relaxations. Additionally, the application of KKT (or subgradient)-based differentiation is limited to convex (or linear) objectives, rendering it not suitable for all CO problems.

3.3 The Statistical Category

The "statistical" category [49, 44] designs loss for PnO considering the statistical relation between multiple solutions. The aim is to make coefficient predictions so that the decision quality (defined in Eq. (4)) of optimal solutions is superior to suboptimal solutions. Representative models include **NCE** [49] and **LTR** [44] (with 3 versions: pointwise-LTR, pairwise-LTR, and listwise-LTR).

The statistical category is usually agnostic to the problem type and solver. However, a solution cache of optimization samples is required before the training, which creates additional overhead. It is also not readily applicable to problem instances of varying variable sizes for solution cache in implementation.

3.4 The Surrogate Category

Another branch besides "statistical" that bypasses gradient estimation is the "surrogate" category. It generally replaces the original optimization objective with a learned surrogate function and has emerged as effective in recent literature. It is inherently differentiable since the objective function is learned. Representative works include **LODL** [60], **EGL** [61], **LANCER** [77], **SurCO** [23], etc.

Similar to the statistical category, the surrogate category easily adapts to all problem and solver types. Furthermore, in cases where solving is slow or assessing solution quality is expensive, using the surrogate expedites the PnO training process. Nevertheless, they do necessitate an extra set of optimization instances and corresponding solutions to learn objective functions as initialization. Additionally, whether and how well they can learn complex nonlinear functions remains open.

3.5 Takeaways for Practitioners' PnO Model Choice

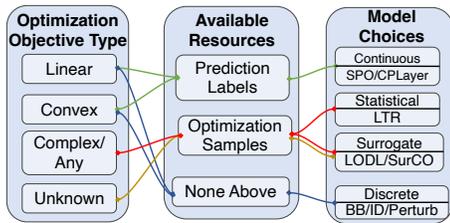



Figure 4: Choice guide of PnO models, which depends on optimization objective type, available resources. From top to bottom, it becomes harder to deploy PnO for harder optimization types and fewer available resources.

Remark on 4 categories of PnO methods: As categorized in Table 1, while end-to-end training on PnO has emerged as a novel paradigm with promising potential to surpass the two-stage PtO paradigm, in practice, there does not currently exist a singular end-to-end model capable of outperforming all other methods: the discrete-category and continuous-category are efficient with fewer solver calls, while constrained by certain optimization forms (e.g. convex), while the latter two categories fit any optimization objectives but with high computational cost and requires additional optimization samples.

PnO model choice guideline As shown in Fig 4, we consider several key factors for the preferable model recommendation: requirement for *labeled prediction datasets*, *extra optimization samples*, and *overall performance* (in both efficacy and efficiency). **Firstly**, the SPO [46] method is preferable for most linear or near-linear problems (Knapsack, energy-cost aware scheduling, budget allocation problem, etc.) as the first try, as long as there are labeled datasets for prediction tasks. SPO requires fewer labeled datasets and is generally

5

Table 2: Dataset statistics. Short terms: integer linear program (ILP), quadratic program (QP).

Name	Data Source	Variable Size	Train Samples	Test Samples	Optimization Type	Used Solver	Prediction Loss
Energy scheduling	SEMO [33]	48	650	139	ILP	Gurobi [28]	MSE
Knapsack (Gen)	Generated [20]	20	400	200	ILP	Gurobi [28]	MSE
Knapsack (Energy)	SEMO [33]	48	400	200	ILP	Gurobi [28]	MSE
Cubic Top-k	Generated [60]	50	250	400	Top-K	Heuristic [51]	MSE
Budget allocation	Yahoo [74]	100	400	200	Submodular	Submodular [37]	BCE
Bipartite matching	Cora [59]	50	20	6	ILP	CVXPY [16]	BCE
Portfolio optimization	Quandl [56]	50	400	200	QP	CVXPY [16]	MSE
Combinatorial Advertising	Industrial	2933(avg)	23	6	ILP	Ortools [53]	BCE

faster regarding additional solving, backward pass complexity, etc. **Secondly**, the LTR [44] method is a good choice for non-linear tasks (TopK, bipartite matching, Portfolio optimization) as long as there are extra optimization samples at hand (or corresponding solution cache), and LODL [60] for better performance if optimization solutions are easy to get (since it requires abundant samples and solutions). **Finally**, the discrete category (Blackbox [54]/Identity [58]) is the last choice for linear/convex problems if there are no labeled datasets for prediction tasks, such as combinatorial advertising.

4 Benchmarks and Remarks

4.1 Modular Framework Design

As shown in Figure 3, we implement the benchmark as a modular framework where each scenario is composed of problems, predictors, solvers, losses, and evaluations. This enables a comprehensive and convenient evaluation of current methods. Although we have not exhaustively addressed all CO problems, we have selected representative ones from real-world applications and introduced a real-world industrial dataset to contribute to the field's development.

4.2 Problem Descriptions

In this section, we briefly introduce each task with the datasets in Table 2, where the background, practical scenarios, detailed "prediction" and "optimization" formulations are left in Appendix C.

4.2.1 Benchmark on publicly available datasets

Energy-cost Aware Scheduling (*SE*) The energy cost-aware scheduling task adopted from [46, 45, 49, 26, 44], is to make machine production schedules based on the predicted energy prices.

Knapsack (*KG*, *KE*) We consider a knapsack problem with unknown item values used in [15, 46, 45, 49, 26]. We provide two versions: the synthetic version [20, 64](knapsack (gen), or "*KG*"), and a real energy dataset version [45, 49, 26] (knapsack (energy), or "*KE*").

Budget Allocation (*BA*) *BA* has applications when agents intend to disseminate information [72, 60] across multiple websites to reach a wider audience within a budget constraint, but the probability of each website reaching users is unknown.

Cubic Top-K (*TK*) The Top-k finds K largest numbers with unknown values following [60].

Bipartite Matching (*BM*) We consider a bipartite graph matching used in [72, 22, 44] where the edge connectivity is unknown and requires prediction.

Portfolio Optimization (*PF*) We introduce portfolio optimization [18, 68] with unknown asset prices, which maximizes the immediate net profit of the securities while reducing risk. Though a continuous quadratic program, we bring it here since it is often regarded as a stress test [60] where the quadratic program naturally provides informative gradients by automatic differentiation [3].

4.2.2 New dataset of combinatorial advertising for inclusive finance

Extending financial services to specific underserved populations within society is highly beneficial in mitigating wealth disparities and enhancing the living standards of low-income individuals. We introduce a new dataset of Combinatorial Advertising ("*CA*") of real industry advertising records, in which a fintech platform connects with financial institutions to provide low-interest loans to users.

6

Table 3: Results for 7 problems on 11 methods. The relative regret (w.r.t. optimal objective %) and average runtime of each epoch are reported. "-" means non-applicable and data generation time of LODL is not counted. Best three: red, orange, blue.

Problem		Predictive	Discrete			Continuous		Statstical				Surrogate
		Two-stage	DFL	Blackbox	Identity	CPLayer	SPO	NCE	point-LTR	pair-LTR	list-LTR	LODL
Knapsack (Gen)	Regret (%)	6.595	11.744	24.274	31.874	24.769	6.223	13.438	6.402	7.820	6.031	6.044
	Train Time	0.101	1.208	2.195	1.351	1.928	1.173	2.065	1.846	4.861	1.536	0.344
	Test Time	0.794	1.636	1.544	1.553	1.928	1.100	1.445	1.682	3.973	1.119	0.716
Knapsack (Energy)	Regret (%)	8.745	8.353	35.705	17.156	36.402	8.407	11.932	8.236	9.022	8.083	9.567
	Train Time	0.198	1.116	1.776	1.051	0.416	2.007	4.010	2.602	10.396	3.099	0.501
	Test Time	0.342	0.760	0.752	0.745	1.825	1.069	2.054	1.406	6.751	1.699	0.344
Scheduling (Energy)	Regret (%)	1.793	6.272	6.503	5.690	-	1.505	1.663	4.548	1.540	1.551	1.786
	Train Time	0.404	51.857	110.647	56.334	-	110.362	109.645	65.744	61.048	61.873	0.582
	Test Time	13.196	26.186	26.799	26.920	-	41.448	37.904	28.590	27.100	26.945	12.055
Budget Allocation	Regret (%)	20.332	35.970	26.905	14.799	-	5.559	9.979	69.663	5.958	5.742	25.700
	Train Time	0.102	4.019	39.799	21.224	-	40.021	7.327	22.669	23.194	22.774	0.248
	Test Time	12.828	25.138	25.728	26.344	-	38.127	38.460	26.392	26.293	26.168	12.278
TopK (Cubic)	Regret (%)	0.110	1.974	13.944	13.944	-	160.408	160.408	1.149	5.072	0.193	0.172
	Train Time	0.064	0.116	0.096	0.097	-	0.126	0.393	0.379	4.653	0.679	0.197
	Test Time	0.038	0.105	0.090	0.087	-	0.125	0.890	0.832	11.625	1.571	0.034
Bipartite Matching	Regret (%)	92.963	91.364	91.988	91.868	92.007	93.327	92.622	91.035	92.285	91.831	91.113
	Train Time	0.010	25.121	0.623	0.334	17.179	1.895	3.467	0.659	1.501	1.531	0.051
	Test Time	0.262	7.725	0.446	0.457	6.643	1.160	1.758	0.886	1.055	1.044	0.239
Portfolio	Regret	0.243	0.380	0.286	0.280	0.309	0.245	0.367	0.214	0.255	0.249	0.160
	Train Time	0.204	0.754	3.762	2.018	1.121	3.446	11.082	3.382	16.383	3.564	0.198
	Test Time	1.187	3.547	2.251	3.115	3.504	3.121	8.071	3.829	17.087	3.919	1.182

Dataset: The advertisement takes place on a mobile application (APP) as a combination of various channels, including in-app notifications, text messages, telephones, etc. Historical data contains whether a user converted after being exposed to a specific marketing combination in the past; however, labels for other combinations remain unknown. The data can be accessed through the website[2], and the processing and use terms are in Appendix C.7.

Prediction: Given user i's feature $\mathbf{x}^i$ (encrypted and processed personal features, app activity records, etc.), predict the user's conversion rate $\mathbf{y}^{ij}$ to the j-th combination of advertising channels.

Optimization: The goal is to allocate the advertiser's existing budget to offer each user a combination that enables broader access (higher conversions) of users to financial services.

It differs from the above *BA* problem as *CA* is based on personalized advertising while *BA* is not. *CA* is also more challenging than the multiple treatment setting [79] due to the exponential combination space of channels, where the latter refers to multiple levels of treatments of the same channel.

4.3 Experimental Setup

We implement our framework based on the previous works [44, 60, 64]. All experiments are carried out on a workstation with Intel® i9-7920X, NVIDIA® RTX 2080, and 128GB RAM.

For the predictive model, we use a two-layer MLP with 32 hidden units in each layer. Unless otherwise specified, we extract 20% from the training data set for validation. During training, we adopt the Adam Optimizer [38] and search by grid the learning rate in $\{0.1, , 0.05, 0.01, 0.005, 0.001\}$. In each run, each model is trained for 300 epochs and the training stops if no better regret is achieved for 50 epochs on the validation set. The epoch that reaches the lowest regret (or highest uplift) in validation is selected for testing. We elaborate on more details of the model design in Appendix B.

4.4 Benchmark Results

We list the benchmarks, including relative regret (with respect to optimal objective) and average runtime for training and test in Table 3, and hyper-parameter sensitivity in Appendix D.3.

From the perspective of each method, methods in the discrete category do not perform as well as others, especially on non-linear optimization tasks (budget allocation, topk, portfolio, etc.). This is probably because the gradient interpolations are only designed for linear objectives and gradient estimations in the back-propagation may not be accurate in complex problems. The continuous-category method SPO performs well on linear problems with relatively short training time. We run CPLayer on the linear optimization problems and non-linear programs are omitted. For the statistical category, listwise-LTR achieves better results than others in most problems, probably because it

---

[2] https://opendata.xinye.com

7

(a) "fine-tuned" BB model (b) "fine-tuned" ID model (c) Prediction loss on training (d) Regret on training

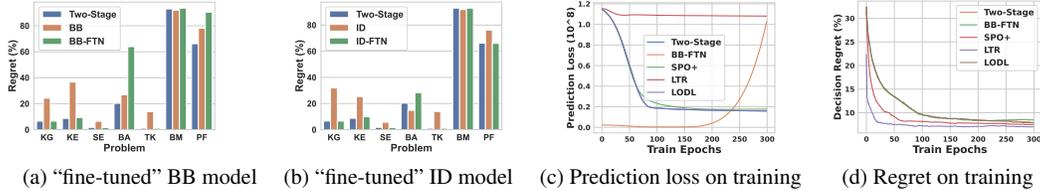



Figure 5: (a~b) Results of fine-tuning (short as "FTN") of the discrete category (BB, ID model). Compared to training PnO directly, "BB-FTN" and "ID-FTN" benefit by first pertaining by PtO and then fine-tuning by PnO on 4,5 over 7 datasets, respectively. (c~d) Learning curve on knapsack (gen) dataset for prediction loss and decision regret w.r.t. training epochs. PnO approaches (LTR, SPO) achieve lower regret than the PtO approach (Two-stage), though with higher prediction loss.

captures the global relationships among multiple solutions. The surrogate method LODL achieves satisfactory results on half of the benchmarks. More analysis is left to Appendix D.1.

As for the training efficiency, one may observe that the two-stage approach is significantly faster than others in the training stage since it does not involve solving the optimization problem. CPLayer may take much longer time than others since the KKT differentiation may incur $O(N^3)$ time complexity for back-propagation. Statistical methods often come with higher training time since multiple solutions are required for model initialization. Though training on LODL is fast, the model preparation time (including collecting solutions for optimization samples and learning the surrogate models for objective functions) is not counted, and it could be time-consuming if the number of required samples is large and solving an optimization problem is slow. We used 5000 samples for LODL, and better LODL performance may occur with more samples (as well as more running time)

Table 4: Combinatorial advertising of "discrete" category.

	Two-stage	DFL	Blackbox	Identity
Uplift	0.069	0.088	0.134	0.135
Train Time	0.170	10.670	10.053	10.120
Test Time	3.078	3.216	3.096	3.199

The result of the combinatorial advertising on discrete category is shown in Table 4, where other categories of methods that cannot run on-the-fly do not apply to this problem directly because they require additional coefficient labels/optimization samples that cannot be satisfied in this problem. We conclude applicability in Sec. 3.5 in the following. The PnO training significantly improves the uplift metric (using the discrete category). The identity model achieves the best result. The DFL and Blackbox approach also performs better than the two-stage approach, though with higher training time. Though the end-to-end PnO training takes higher training time, they do not incur additional runtime during the test stage.

Remark of PnO methods by empirical results: In the benchmark results, we observe that in line with analysis in Sec. 3, despite the immediate applicability of discrete-category methods without extra data or solving, their reliance on interpolated gradients often leads to inaccuracies during training, resulting in inferior performance compared to the two-stage methods in many scenarios. Statistical-category and surrogate-category methods exhibit better performance. However, it is noteworthy that they incur higher computational costs during data sample collection and forward-pass computations.

The above categorizations and benchmark suggest that *PnO methods lack a universally accepted standard, and in practical applications, each method has its own advantages and disadvantages*. Therefore, there is an urgent need for better approaches with respect to applicability, efficiency and performance. Hence, addressing the imperative challenge of attaining a more **universal**, **stable**, and **efficient** training model is essential to fully realize the potential of "PnO" in the real world.

4.5 Key Factors for PnO Methodology Design

To investigate the key factors for future improvements of PnO, we propose to explore the following research questions: (**RQ1**) What is the relationship between prediction accuracy and decision quality in the PnO methods? (**RQ2**) To what extent do the prediction labels impact the PnO methods? (**RQ3**) Do the PnO methods demonstrate versatility across different settings?

RQ1: Relationships of prediction accuracy and decision quality In common sense, a better prediction would lead to a better subsequent decision. Besides a case study shown in [20], we explore

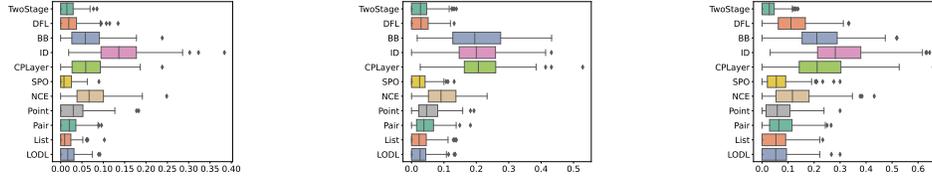
8

(a) Variable size 20, Capacity 90   (b) Variable size 20, Capacity 60   (c) Variable size 20, Capacity 30

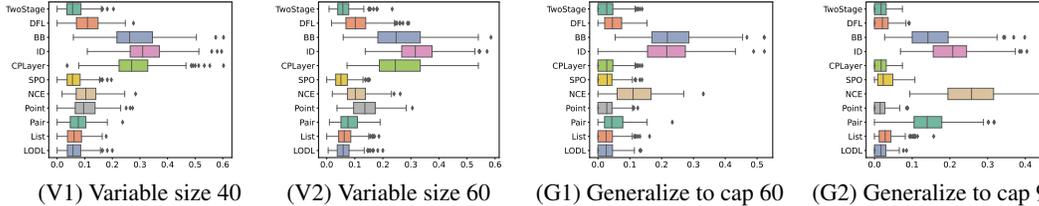
Figure 6: Regret on knapsack (gen) problem. The decision degrades with tighter constraints.

(V1) Variable size 40   (V2) Variable size 60   (G1) Generalize to cap 60   (G2) Generalize to cap 90

Figure 7: Regret on knapsack (gen) problem. (V1)~(V2) demonstrates performance with different variable sizes, and (G1)~(G2) generalizes the trained model on capacity 30 to 60, 90 on the test data. Most PnO models degrade when the test set distribution shifts and are less stable than the two-stage approach with different variable sizes. Detailed results in Table. 7.

the prediction loss and decision objective in both PtO and PnO. We illustrate the learning curve on the real-world knapsack (energy) dataset by plotting the prediction loss (Mean-squared error) and evaluation results (regret) of one method for each category, shown in Figure 5(c d).

It is observed that PnO training methods like SPO, listwise-LTR, and LODL achieve lower regret than the vanilla PtO method. The prediction loss of LODL shares a similar pattern with PtO, while LTR's is different. This demonstrates despite higher prediction loss incurred in the end-to-end training, these PnO methods obtain better decisions by their own loss functions concerning the ultimate decision objectives. This might correspond to the proposition [9] that *PnO achieves better decision quality by finding better error trade-offs by information of decision objectives*. We show more qualitative analysis in Appendix D.5.

RQ2: Impact of prediction labels on PnO We explore the impacts of prediction labels specifically for discrete categories (Blackbox and Identity) since they inherently do not use label information. Without the true prediction label $\mathbf{Y}$, training from scratch by respective PnO loss could make it very slow to converge, as a possible reason for demonstrating inferior performance in Table 3. We attempt to expose the model with true prediction labels as a "warm-up" before the PnO training: we initially train for 150 epochs using a PtO MSE loss, followed by 150 epochs of fine-tuning using the PnO loss. The results are denoted as "BB-FTN" and "ID-FTN" respectively, where "FTN" is short for fine-tuning. It could be observed from Figure 5(a b) that this hybrid training on Blackbox and Identity method demonstrates improvements over trained directly by PnO across 4 and 5 over 7 datasets, respectively, and surpasses the two-stage method in some cases. This may indicate that *supervised PtO pretraining could be beneficial for PnO models, especially for the discrete category that inherently does not rely on prediction labels* (see Table 1 for requirements for prediction labels).

RQ3: Versatility of PnO under various parameters We explore the versatility of PnO as follows. (**1**) Figure 6 examines the impact of varying constraint on the training of *KG* dataset. We set the capacity of 30, 60, and 90. It was observed that as the constraints became more restrictive, the relative regrets became higher. This observation potentially indicates that constraint satisfaction is a prevalent factor constraining the performance of current PnO training methods. (**2**) Figure 7(V1~V2) show the performance when the number of decision variables increases to 40 and 60. With increasing variable size, the two-stage approach remains stable under different problem settings, though its final decision quality is inferior to some of the PnO training models. Conversely, some PnO models exhibited fluctuations and degradation, such as discrete-category methods and point-wise LTR. (**3**) Figure 7(G1~G2) show the generalizability when the PnO models are trained on one problem setting (e.g., constraint of capacity 30) and tested on the other (e.g., constraint of capacity 60 or 90). Similar performance degradation occurs in some PnO models, including discrete-category and statistical-category models, in this setting. This indicates *current PnO approaches may perform inadequately when facing more stringent constraints, larger decision variable sizes, and the need for generalization across diverse optimization parameters*. More details are shown in Appendix D.2.

9

5 Conclusion

Regarding predictive CO problems, we systematically review existing PtO/PnO methods, implement a framework for comprehensive assessment, and perform experiments to explore scenarios in which PnO methods could surpass the PtO approaches. Furthermore, we introduce a novel dataset, which will be publicly available along with the framework. We hope to facilitate the community and industry in swiftly reproducing, developing, and deploying new algorithms in the real world.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. *Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[2] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a cone program. *arXiv preprint arXiv:1904.09043*, 2019.

[3] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, pages 136–145. PMLR, 2017.

[4] E. Balkanski and Y. Singer. Minimizing a submodular function from samples. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[5] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[6] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with differentiable perturbed optimizers. *Neural Information Processing Systems (NeurIPS)*, 33:9508–9519, 2020.

[7] D. Bertsimas and N. Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

[8] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021.

[9] C. Cameron, J. Hartford, T. Lundy, and K. Leyton-Brown. The perils of learning before optimizing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 3708–3715, 2022.

[10] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine Learning (ICML)*, pages 129–136, 2007.

[11] R. Caruana, S. Baluja, and T. Mitchell. Using the future to" sort out" the present: Rankprop and multitask learning for medical risk evaluation. *Neural Information Processing Systems (NIPS)*, 8, 1995.

[12] J. Chen, J. Wang, Z. Zhang, Z. Cao, T. Ye, and S. Chen. Efficient meta neural heuristic for multi-objective combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] W. Chen, X. Sun, J. Zhang, and Z. Zhang. Optimization from structured samples for coverage functions. In *International Conference on Machine Learning (ICML)*, pages 1715–1724. PMLR, 2020.

[14] X. Chen, Y. Li, R. Wang, and J. Yan. Mixsatgen: Learning graph mixing for sat instance generation. In *The Twelfth International Conference on Learning Representations*, 2024.

10

[15] E. Demirović, P. J. Stuckey, J. Bailey, J. Chan, C. Leckie, K. Ramamohanarao, and T. Guns. An investigation into prediction+ optimisation for the knapsack problem. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4–7, 2019, Proceedings 16*, pages 241–257. Springer, 2019.

[16] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[17] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] P. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In *Neural Information Processing Systems (NIPS)*, pages 5484–5494, 2017.

[19] M. Eisenberger, A. Toker, L. Leal-Taixé, F. Bernard, and D. Cremers. A unified framework for implicit sinkhorn differentiation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 509–518, 2022.

[20] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

[21] A. N. Elmachtoub, J. C. N. Liang, and R. McNellis. Decision trees for decision-making under the predict-then-optimize framework. In *International Conference on Machine Learning (ICML)*, pages 2858–2867. PMLR, 2020.

[22] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 1504–1511, 2020.

[23] A. M. Ferber, T. Huang, D. Zha, M. Schubert, B. Steiner, B. Dilkina, and Y. Tian. Surco: Learning linear surrogates for combinatorial nonlinear optimization problems. In *International Conference on Machine Learning (ICML)*, pages 10034–10052. PMLR, 2023.

[24] J. Futoma, M. Hughes, and F. Doshi-Velez. Popcorn: Partially observed prediction constrained reinforcement learning. In *International Conference on Artificial Intelligence and Statistics(AISTATS)*, pages 3578–3588. PMLR, 2020.

[25] D. Goldenberg, J. Albert, L. Bernardi, and P. Estevez. Free lunch! retrospective uplift modeling for dynamic promotions recommendation within roi constraints. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 486–491, 2020.

[26] A. U. Guler, E. Demirović, J. Chan, J. Bailey, C. Leckie, and P. J. Stuckey. A divide and conquer algorithm for predict+ optimize with non-convex problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 3749–3757, 2022.

[27] Z. Guo, Y. Li, C. Liu, W. Ouyang, and J. Yan. Acm-milp: Adaptive constraint modification via grouping and selection for hardness-preserving milp instance generation. In *The Forty-first International Conference on Machine Learning*, 2024.

[28] G. O. Gurobi. Llc.,". *Gurobi Optimizer Reference Manual*, 2019.

[29] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics (AISTATS)*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[30] K. Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, pages 24–50, 2017.

[31] C. Hertrich and M. Skutella. Provably good solutions to the knapsack problem via neural networks of bounded size. *INFORMS journal on computing*, 35(5):1079–1097, 2023.

11

[32] M. Hughes, G. Hope, L. Weiner, T. McCoy, R. Perlis, E. Sudderth, and F. Doshi-Velez. Semi-supervised prediction-constrained topic models. In *International Conference on Artificial Intelligence and Statistics(AISTATS)*, pages 1067–1076. PMLR, 2018.

[33] G. Ifrim, B. O'Sullivan, and H. Simonis. Properties of energy-price forecasts for scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 957–972. Springer, 2012.

[34] Y. Jin, Y. Ding, X. Pan, K. He, L. Zhao, T. Qin, L. Song, and J. Bian. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pages 8132–8140, 2023.

[35] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[36] W. Joe and H. C. Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 394–402, 2020.

[37] M. Karimi, M. Lucic, H. Hassani, and A. Krause. Stochastic submodular maximization: The case of coverage functions. *Advances in Neural Information Processing Systems*, 30, 2017.

[38] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[39] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[40] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 10657–10665, 2019.

[41] Y. Li, X. Chen, W. Guo, X. Li, W. Luo, J. Huang, H.-L. Zhen, M. Yuan, and J. Yan. Hardsatgen: Understanding the difficulty of hard sat formula generation and a strong structure-hardness-aware baseline. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2023.

[42] Y. Li, J. Guo, R. Wang, and J. Yan. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[43] Y. Li, J. Guo, R. Wang, H. Zha, and J. Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[44] J. Mandi, V. Bucarey, M. M. K. Tchomba, and T. Guns. Decision-focused learning: through the lens of learning to rank. In *International Conference on Machine Learning (ICML)*, pages 14935–14947. PMLR, 2022.

[45] J. Mandi and T. Guns. Interior point solving for lp-based prediction+ optimisation. *Neural Information Processing Systems (NeurIPS)*, 33:7272–7282, 2020.

[46] J. Mandi, P. J. Stuckey, T. Guns, et al. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 1603–1610, 2020.

[47] H. M. Markowitz and G. P. Todd. *Mean-variance analysis in portfolio choice and capital markets*, volume 66. John Wiley & Sons, 2000.

[48] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

12

[49] M. Mulamba, J. Mandi, M. Diligenti, M. Lombardi, V. Bucarey, and T. Guns. Contrastive losses and solution caching for predict-and-optimize. In Z.-H. Zhou, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2833–2840. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[50] M. Niepert, P. Minervini, and L. Franceschi. Implicit mle: backpropagating through discrete exponential family distributions. *Neural Information Processing Systems (NeurIPS)*, 34:14567–14579, 2021.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[52] A. Paulus, M. Rolínek, V. Musil, B. Amos, and G. Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning (ICML)*, pages 8443–8453. PMLR, 2021.

[53] L. Perron and V. Furnon. Or-tools, 2022-11-25.

[54] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

[55] R. Qiu, Z. Sun, and Y. Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Neural Information Processing Systems (NeurIPS)*, 35:25531–25546, 2022.

[56] Quandl. Wiki various end-of-day data, 2022.

[57] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[58] S. S. Sahoo, A. Paulus, M. Vlastelica, V. Musil, V. Kuleshov, and G. Martius. Backpropagation through combinatorial algorithms: Identity with projection works. In *International Conference on Learning Representations (ICLR)*, 2023.

[59] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[60] S. Shah, K. Wang, B. Wilder, A. Perrault, and M. Tambe. Decision-focused learning without decision-making: Learning locally optimized decision losses. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Neural Information Processing Systems (NeurIPS)*, 2022.

[61] S. Shah, B. Wilder, A. Perrault, and M. Tambe. Leaving the nest: Going beyond local loss functions for predict-then-optimize. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14902–14909, 2024.

[62] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[63] Z. Sun and Y. Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[64] B. Tang and E. B. Khalil. Pyepo: A pytorch-based end-to-end predict-then-optimize library for linear and integer programming. *arXiv preprint arXiv:2206.14234*, 2022.

[65] G. Tong. Usco-solver: Solving undetermined stochastic combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 34:1646–1659, 2021.

[66] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[67] D. Wahdany, C. Schmitt, and J. L. Cremer. More than accuracy: end-to-end wind power forecasting that optimises the energy system. *Electric Power Systems Research*, 221:109384, 2023.

13

[68] K. Wang, B. Wilder, A. Perrault, and M. Tambe. Automatically learning compact quality-aware surrogates for optimization problems. *Advances in Neural Information Processing Systems*, 33:9586–9596, 2020.

[69] R. Wang, L. Shen, Y. Chen, X. Yang, D. Tao, and J. Yan. Towards one-shot neural combinatorial solvers: Theoretical and empirical notes on the cardinality-constrained case. In *International Conference on Learning Representations (ICLR)*, 2023.

[70] R. Wang, J. Yan, and X. Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pages 3056–3065, 2019.

[71] R. Wang, Y. Zhang, Z. Guo, T. Chen, X. Yang, and J. Yan. Linsatnet: the positive linear satisfiability neural networks. In *International Conference on Machine Learning (ICML)*, pages 36605–36625. PMLR, 2023.

[72] 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 (AAAI)*, volume 33, pages 1658–1665, 2019.

[73] Y. Xie, H. Dai, M. Chen, B. Dai, T. Zhao, H. Zha, W. Wei, and T. Pfister. Differentiable top-k with optimal transport. *Neural Information Processing Systems (NeurIPS)*, 33:20520–20531, 2020.

[74] Yahoo. Yahoo! webscope dataset, 2007.

[75] K. Yan, J. Yan, C. Luo, L. Chen, Q. Lin, and D. Zhang. A surrogate objective framework for prediction+programming with soft constraints. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 21520–21532. Curran Associates, Inc., 2021.

[76] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[77] A. Zharmagambetov, B. Amos, A. Ferber, T. Huang, B. Dilkina, and Y. Tian. Landscape surrogate: Learning decision losses for mathematical optimization under partial information. *Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[78] C. Zhou, J. Ma, L. Douge, E. P. Chew, and L. H. Lee. Reinforcement learning-based approach for dynamic vehicle routing problem with stochastic demand. *Computers & Industrial Engineering*, 182:109443, 2023.

[79] H. Zhou, S. Li, G. Jiang, J. Zheng, and D. Wang. Direct heterogeneous causal learning for resource allocation problems in marketing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pages 5446–5454, 2023.

14

A   Related Work

Despite the significant progress neural networks have made in machine learning, there is still substantial room for improvement when it comes to solving optimization problems. Although several studies [55, 34, 63, 12, 31, 42, 43, 41, 14, 27] have explored the use of neural networks for combinatorial optimization problems in deterministic environments, especially in the constrained CO tasks [71, 69], however, many of them are incompetent to catch up heuristic algorithms (such as the LKH for the traveling salesman problem) in terms of solving efficiency and decision quality. More importantly, they are not directly applicable to the optimization tasks under uncertainty, especially for unknown coefficients. Therefore, a newer area of interest involves combining machine learning with optimization techniques to tackle problems with uncertain parameters, elaborated as follows.

[7] pioneered efforts in merging predictive and prescriptive analysis for optimization under uncertainty. A notable contribution is the SPO method [20], which introduces subgradient-based surrogate functions to replace non-differentiable regret functions in linear optimization problems, and its extension, SPO-relax [46], which applies continuous relaxation to combinatorial problems and is utilized in our experiments. It is important to note that these studies primarily focus on predictive models before optimization solvers, which are often treated as standard heuristics (such as LKH3 [30] or Dijkstra [17]) or commercial solvers (like Gurobi [28]).

Recently, there has been a growing body of work on predict-and-optimize approaches (also known as decision-focused learning [72, 44, 60]). A significant trend within this category involves using relationships between solutions to optimization problems to develop better predictive models. For instance, the NCE (Noise-Contrastive Estimation) method [49] utilizes a noise-contrastive estimation technique [29] to produce predictions that aim to achieve optimal solutions with better decision quality compared to non-optimal ones. The LTR (Learning to Rank) technique [44] explores the connection between pairwise learning to rank in NCE, leading to the development of various learning-to-rank methodologies such as pointwise rank [11], pairwise rank [35], and listwise rank [10], which aim to generate predictions reflecting the relative importance of multiple solutions.

Another recent approach involves using neural network functions as surrogates for the original objective functions. Studies such as LODL [60] and EGL [61] propose learning surrogate objective functions from sample data. LANCER [77] adopts a similar strategy, integrating surrogate function learning with optimization solving and objective function learning. SurCO [23] suggests replacing the original non-linear objective with a linear surrogate, allowing the use of existing linear solvers.

In addition, some works propose implicit differentiation to obtain derivatives for convex optimization objectives for continuous optimization problems. OptNet [3] is the first work to differentiate quadratic programs, and then CPLayer [1] and more [18, 40, 19] extends this spirit to more general convex programs. QPTL [72] proposes to differentiate through combinatorial optimization tasks by relaxing the problem to the continuous pace, and IntOpt [45] improves the interior-point method so that its backpropagation differentiable. The following works [68, 22, 52] generate this paradigm for discrete problems.

However, some of these methods are limited to specific types of predict-and-optimize problems, such as quadratic optimization objectives [3] or convex objectives. Although methods based on solution importance [49, 44] and surrogate objective functions [60, 61, 77] do not restrict the type of optimization, they require additional information, such as multiple solutions or a large number of optimization samples [60], to train the surrogate function before complete end-to-end learning can be performed. Furthermore, **the most critical issue** is that most of these methods struggle with combinatorial optimization due to the difficulty of differentiating through discrete decision variables, making predict-and-optimize for CO problems particularly challenging.

Besides the works mentioned above, some works also focus on the optimization under uncertainty. However, since current PnO methods do not support arbitrary uncertainty scenarios and are limited to cases with a known closed-form optimization objective but unknown coefficients, they are currently out of the scope of our benchmark. In [36, 78], the form of the optimization objective function is uncertain and changes with time (especially for the pickup task), and [65] focuses more on the performance of regression among combinatorial spaces when there is no distribution of test data during training, but training data generation is available. Therefore, in this work, we mainly focus on methods within the predict-and-optimize paradigm and do not include them for now.

15

B Model Details

B.1 Prediction Model

For a fair comparison, all end-to-end training models, as well as the two-stage approach, utilize the same multi-layer perception (MLP) for the prediction of optimization coefficients. The prediction model $\mathcal{M}$ using MLP is:

$$\mathbf{a}^{(i+1)} = \sigma(\mathbf{W}^{(i)}\mathbf{a}^{(i)} + \mathbf{b}^{(i)}), \quad i = 1, 2, \ldots, K-1, \tag{6}$$

where $\mathbf{a}^{(1)} = \mathbf{x}$ and $\mathbf{y} = \mathbf{a}^{(K)}$ are input and output for $\mathcal{M}$, a^i is the hidden vector for $i = 2, \cdots, K-1$, $\mathbf{W}$ is the weight term, b is the bias term and σ is the activation function where we adopt ReLU. In experiments, we adopt $K = 3$ layers and the size of intermediate hidden units is set as 32.

B.2 Two-stage approach in PtO

The two-stage approach is the vanilla model used when the coefficients of the optimization problem are uncertain and require prediction. In the prediction stage, the predictor estimates the coefficients of the optimization problem and optimizes with the original prediction loss using pre-collected labeled data pairs $D = \{x_i, y_i\}$. In the optimization stage, the coefficient on the test data is first inferred so that the optimization problem becomes certain. The vanilla solver is used to solve the optimization problem.

The final loss used in the two-stage approach is the same as the prediction stage, which includes the mean squared error (MSE) loss or binary cross entropy (BCE) loss of the prediction stage.

$$\begin{aligned}
\text{MSE}(\hat{y}, y) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \\
\text{BCE}(\hat{y}, y) &= -(y\log(a^{(K)}) + (1-y)\log(1 - a^{(K)})),
\end{aligned} \tag{7}$$

where $a^{(K)}$ is the output logits for the final layer neural network, n is the number of training samples in MSE. The specific loss selection depends on whether the prediction task is regression (with MSE) or binary classification (with BCE). The used final loss is listed in Table 2.

B.3 Details for PnO models

B.3.1 The Discrete category

Table 5 lists the gradient interpolations for the discrete category.

A simple solution is to directly pass the gradient of the predicted coefficients from the objective, namely decision-focused learning (**DFL**) following [60]. This spirit is also present in the straight-through estimator (**STE**) [5] to estimate the gradient of the ReLU activation function. The representative work by gradient interpolation is **Blackbox** [54] which proposes linear interpolation by conducting a minor perturbation of original coefficient $\hat{y}$ to produce informative gradients. More recently proposed **Perturb** [6] and **I-MLE** [50] adopt the same spirit to obtain informative gradients, and **Identity** [58] ignores the constraints and treats the gradient as a negative identity matrix.

B.3.2 The Continuous category

Popular approaches use the differentiation of Karush-Kuhn-Tucker (KKT) conditions proposed in **OptNet** [3] to obtain $\partial \mathbf{z}/\partial \mathbf{y}$ for the quadratic program. The later works extend this approach to cone programs [2] (**cvxpy layer**) and more [18, 40, 19]. However, these methods do not consider the case for discrete variables where gradients are blocked. Several studies [72, 45, 68, 22, 52] have extended the KKT differential approach to discrete problems. **QPTL** [72] extends this branch further to address singular gradient issues in discrete linear optimization by adding a squared norm regularizer. **IntOpt** [45] addresses the challenge by differentiating homogeneous self-dual formulation for mixed integer programs to overcome difficulties of constraint satisfaction in computing the gradients using KKT conditions.

16

Table 5: Gradient interpolations for "discrete" category where λ is hyperparameter in Blackbox controlling trade-off between "faithfulness" and "informativeness", $\mathbf{I}$ is identity matrix, $\tau > 0$ is temperature and R is a sampled noise vector.

Method	Surrogate gradient
Blackbox [54]	$\frac{\partial L}{\partial \mathbf{y}} = \frac{1}{\lambda}\left[\mathbf{z}(\hat{\mathbf{y}} + \lambda\frac{\partial L}{\partial \mathbf{z}}(\hat{\mathbf{z}})) - \mathbf{z}(\hat{\mathbf{y}})\right]$
Identity [58]	$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = -\mathbf{I}$
Perturb-and-MAP [6]	$\frac{\partial L}{\partial \mathbf{y}} = \mathbf{z}(\hat{\mathbf{y}} + \tau R) - \mathbf{z}(\mathbf{y})$
I-MLE [50]	$\frac{\partial L}{\partial \mathbf{y}} = \mathbf{z}(\hat{\mathbf{y}} + \lambda\frac{\partial L}{\partial \mathbf{z}}(\hat{\mathbf{z}}) + R) - \mathbf{z}(\hat{\mathbf{y}} + R)$

Table 6: Loss functions for "statistical" category, where $p_\tau(\mathbf{z} \mid \mathbf{y}) = \exp(-\frac{f(\mathbf{z},\mathbf{y})}{\tau})/\Sigma_{\mathbf{z}' \in S}\exp(\frac{-f(\mathbf{z}',\mathbf{y})}{\tau})$, (p,q) is in ordered pairs OP means $f(\mathbf{v}_p, \mathbf{y}) < f(\mathbf{v}_q, \mathbf{y})$, the positive parameter ν is the margin, and S represents the set of feasible solutions.

Method	Designed loss function
NCE [49]	$\Sigma_{\mathbf{z}_s \in S}(f(\mathbf{z}^\star, \hat{\mathbf{y}}) - f(\mathbf{z}_s, \hat{\mathbf{y}}))$
pointwise-LTR [44]	$\frac{1}{|S|}\Sigma_{\mathbf{z}_s \in S}(f(\mathbf{z}_s, \hat{\mathbf{y}}) - f(\mathbf{z}_s, \mathbf{y}))^2$
pairwise-LTR [44]	$\Sigma_{(p,q)\in OP}(\nu + f(\mathbf{z}_p, \hat{\mathbf{y}}) - f(\mathbf{z}_q, \hat{\mathbf{y}}))$
listwise-LTR [44]	$\frac{1}{|S|}\Sigma_{\mathbf{z}_s \in S}p_\tau(\mathbf{z}_s \mid \mathbf{y})(\log p_\tau(\mathbf{z}_s \mid \mathbf{y}) - \log P_\tau(\mathbf{z}_s \mid \hat{\mathbf{y}}))$

Alternatively, another branch of research [46, 21] has focused on adapting subgradient approximation methods from continuous linear problems to discrete scenarios. **SPO-relax** [46] relaxes the problem and utilizes the surrogate **SPO+** loss proposed in [20], and use the subgradient for backpropagation.

$$L_{spo}(\mathbf{y}, \hat{\mathbf{y}}) = -f(\mathbf{z}^\star(2\hat{\mathbf{y}} - \mathbf{y}), 2\hat{\mathbf{y}} - \mathbf{y}) + 2f(\mathbf{z}^\star(\mathbf{y}), \hat{\mathbf{y}}) - f(\mathbf{z}^\star(\mathbf{y}), \mathbf{y}). \tag{8}$$

The above categories propose on-the-fly gradient estimates to make gradients accessible. On the contrary, the subsequent two categories bypass the gradient approximation to train PnO.

B.3.3 The Statistical category

Table 6 lists the designed loss functions for the statistical category.

The representative method **NCE** [49] designs a noise-contrastive estimation (NCE) [29] approach to produce predictions such that optimal solutions gain better decision quality than non-optimal ones. **LTR** [44] discovers the inherent correlation between pairwise learning to rank and NCE, leading to the proposition of a series of learn-to-rank methodologies, including pointwise rank [11], pairwise rank [35], and listwise rank [10] loss to produce predictions that capture the relative importance of multiple solutions.

B.3.4 The Surrogate Category

Recent works **LODL** [60] and **EGL** [61] propose learning the surrogate of objective functions from a set of samples. **LANCER** [77] learns surrogate functions in a similar way, while it integrates optimization solving and learning of objective functions. **SurCO** [23] proposes to replace the original non-linear objective with a linear surrogate, thus enabling the utilization of existing linear solvers.

C Problem and Dataset Details

For most problems, we adopt generated data or data that has been publicly available. For our new benchmark combinatorial advertising problem, which will be released, we collect data that contain encrypted and processed personal features and mobile application activity records, and all the used data are permitted by the users. In the following, we elaborate on the detailed data processing procedure for each problem. We also give code framework ?? with base classes.

17

Table 7: Sensitivity analysis on capacity, variable size, and generalization on capacity on Knapsack (Gen) evaluated by regret(Best three: red, orange, blue.

Setting		Predictive Two-stage	Discrete DFL	Blackbox	Identity	Continuous CPLayer	SPO+	Statistical NCE	point-LTR	pair-LTR	list-LTR	Surrogate LODL
Sensitivity on Capacity (Figure. 6)	Capacity 30	6.595	11.744	24.274	31.874	24.769	6.223	13.438	6.402	7.820	6.031	6.044
	Capacity 60	3.188	3.355	21.244	22.824	23.588	2.933	10.459	5.480	4.488	2.960	3.030
	Capacity 90	1.566	2.499	7.325	14.985	7.511	1.345	7.945	3.901	2.221	1.417	1.773
Sensitivity on variable size (Figure. 8)	Size 40	6.242	11.480	29.934	34.519	29.835	5.940	11.178	11.209	8.193	6.506	6.408
	Size 60	5.941	10.134	25.719	32.576	26.475	5.675	10.598	13.980	8.377	7.056	5.996
	Size 80	5.828	12.241	27.149	36.901	27.324	5.438	10.927	14.080	7.546	6.328	6.159
	Size 100	6.298	10.280	25.954	30.501	26.454	6.055	11.328	15.949	7.593	7.115	6.380
Generalization on capacity (Figure. 9)	30-> 60	3.197	4.848	23.913	25.046	3.197	3.197	12.312	2.995	5.331	3.091	3.065
	30->90	1.716	2.214	15.790	21.919	1.716	2.513	25.970	1.694	14.010	3.047	1.723
	30->120	0.644	0.303	8.724	30.709	0.644	7.151	34.304	0.354	23.646	10.727	0.152

C.1 Energy-cost aware Scheduling

With the more adoption of clean energy sources, energy demand and prices have become more adaptable [57]. In the context of industrial production, optimizing scheduling tasks based on energy prices has the potential to significantly conserve energy and reduce operational costs.

Prediction: The task is to predict the energy price of the 48 time slots (30 minutes each) given relevant features, including weather estimates, temperature, wind energy production, etc.

Optimization: The objective is to minimize the total energy cost of the schedule when the J jobs are scheduled on M machines and abide by the earliest start time and latest end time constraint.

Suppose there are M machines to deploy J jobs with R resources, and each day is split in T equal timeslots ($T = 48$ in our case). Each job comes with its earliest start time e_j, latest end time l_j, duration d_j and power usage p_j. u_{jr} is resource usage of task j on resource r. c_{mr} denotes the capacity of machine m for resource r. The optimization objective is to minimize a linear program that completes scheduling tasks and reduces the energy cost:

$$\min_{\mathbf{z}_{jmt}} \Sigma_{j \in J} \Sigma_{m \in M} \Sigma_{t \in T} \mathbf{z}^{jmt} \left(\Sigma_{t \leq t' < t + d_j} p^j \mathbf{y}^{t'} \right) \tag{9}$$

$$\Sigma_{m \in M} \Sigma_{t \in T} \mathbf{z}^{jmt} = 1, \forall_{j \in J} \tag{10}$$

$$\mathbf{z}^{jmt} = 0 \quad \forall_{j \in J} \forall_{m \in M} \forall_{t < e_j} \tag{11}$$

$$\mathbf{z}^{jmt} = 0 \quad \forall_{j \in J} \forall_{m \in M} \forall_{t + d_j > l_j} \tag{12}$$

$$\Sigma_{j \in J} \Sigma_{t - d_j < t' \leq t} \mathbf{z}^{jmt'} u_{jr} \leq \mathbf{y}^{mr}, \forall_{m \in M} \forall_{r \in R} \forall_{t \in T} \tag{13}$$

where $\mathbf{z}^{jmt}$ is a binary decision variable which is 1 if task j starts at time t on machine m, else 0. Constraint (10) ensures each task is scheduled once and only once. *Constraint* (11) and *Constraint* (12) make sure the job starts after the earliest start time and ends before the latest end time, and *Constraint* (13) indicates the job does not exceed the capabilities of machines. The data of each day is treated as one optimization instance. We adopt $N = 3$ machines, $R = 1$ resources. The resource usage u_{jr} is certain and given.

Dataset: The data adopts open-sourced Irish Single Electricity Market Operator (SEMO) [33] collected from Midnight 1st November 2011 to 31st December 2013.

The price that needs to be predicted at each timeslot includes a 9-dimension vector as the input feature: calendar attributes, day-ahead estimates of weather characteristics, SEMO day-ahead forecasted energy load, wind-energy production and prices, and actual wind speed, temperature, CO_2 intensity, and price.

License: We adopt the publicly available dataset SEMO[3], which is licensed[4] and regulated cooperatively by the Commission for Regulation of Utilities (CRU) in Ireland and the Utility Regulator for Northern Ireland (UREGNI, previously named NIAUR).

---

[3]https://www.sem-o.com/about/
[4]https://www.sem-o.com/web-accessibility/

18

C.2 Knapsack

The knapsack problem under uncertainty finds its applications, such as when an agent aims to efficiently carry items while reducing transportation costs and energy consumption, where the energy cost is unknown, which describes the following knapsack (energy) problem.

Prediction: The task is to predict the j-th item's value $\mathbf{y}^j$ from feature vector $\mathbf{x}^j$ for each N item.

Optimization: The optimization task is to maximize the total value of items in the knapsack with a capacity constraint, formulated as an integer linear objective:

$$\mathbf{z}^{\star}(y) = \arg\max_{\mathbf{z}} \Sigma_{j=1}^{N} \mathbf{y}^j \mathbf{z}^j \quad \text{s.t. } \Sigma_{j=1}^{N} \mathbf{w}^j \mathbf{z}^j \leqslant C . \tag{14}$$

where $\mathbf{z}^j \in \{0, 1\}$ is the binary variable indicating whether the item is picked, $\mathbf{w}^j$ is the weight for each item, and C is the total capacity.

Dataset:

- **Synthetic** A synthetic dataset $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ is generated by the polynomial function following previous literature [20]:

$$\mathbf{y}_i = [\frac{1}{3.5^{\deg}\sqrt{p}} ((\mathcal{B}\mathbf{x}_i) + 3)^{\deg} + 1] \cdot \epsilon_i, \tag{15}$$

  where each $x_i \sim N(0, I_p)$ is generated from a multivariate Gaussian distribution, matrix ($\mathcal{B}^* \in \mathbb{R}^{d \times p}$ encodes the parameters of the true model, whereby each entry of $\mathcal{B}^*$ is a Bernoulli random variable that is equal to 1 with probability 0.5, ϵ_i^j is a multiplicative noise term with uniform distribution, and p is given number of features. The weights of the knapsack problem are regarded as certain and are sampled uniformly between 3 and 8. In our experiments, we set the default capacity as 30, and the number of items is 20. We use the polynomial degree $\deg$ of 4.

- **Real** The real data for knapsack adopts the SEMO dataset [33][3] in the previous energy-cost aware problem, where the energy price of each time slot is seen as profit, and the resource usage u_{jr} of each time slot is seen as the weight of each item. which is certain and given.

C.3 Budget Allocation

The budget allocation problem under uncertainty has applications in scenarios where nonprofits aim to disseminate philanthropic information across multiple websites within the constraints of a total budget.

Prediction: Given the features $\mathbf{x}^w$ associated with the website w, the task is to predict $\mathbf{y}^w$, the probability that the information on the website i will reach the customers.

Optimization: The objective is to maximize the expected number of users that are reached by the website at least once:

$$\mathbf{z}^{\star}(\mathbf{y}) = \arg\max_{\mathbf{z}} \frac{1}{N} \Sigma_{u=0}^{N} \left(1 - \prod_{w=0}^{M} (1 - \mathbf{z}^w \cdot \mathbf{y}^{wu}) \right) \quad \text{s.t. } \Sigma_{w=0}^{M} \mathbf{z}^w \leqslant B . \tag{16}$$

where N is the number of users and M is the number of websites.

Dataset: The data comes from Yahoo! Webscope Dataset [74] with labels y_i for each user i. We adopt the multi-linear relaxation following previous literature [72] where the features are generated by multiplying a random matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, i.e. $\mathbf{x}_i = \mathbf{A}\mathbf{y}_i$. In our experiments, we set 5 websites, and 10 users with the default budget 1 following [72, 60].

License: We obtain this dataset from Yahoo's publicly available data [5], which is intended for non-commercial use by academics and other researchers. This data complies with Yahoo's data protection standards [6], including stringent privacy controls. Usage of the data must adhere to the Data Sharing Agreement and terms of use [7] by Yahoo.

---

[5]https://webscope.sandbox.yahoo.com
[6]http://privacy.yahoo.com
[7]https://legal.yahoo.com/us/en/yahoo/terms/otos/index.html

19

C.4 Cubic Top-K

The top-k problem adopted from [60] finds application in the field of Explainable XAI literature [32, 24]. Here, the task revolves around the identification of the most salient features within the predictive model, specifically denoted as top-k.

Prediction: Given a dataset $\{\mathbf{x}_i, \mathbf{y}_i\}$, where $\mathbf{x}_i \sim U[0, 1]$ is sampled from a uniform distribution, and $\mathbf{y}_i$ is generated according to cubic polynomial as follows:

$$\mathbf{y}_i = 10\mathbf{x}_i^3 - 6.5\mathbf{x}_i . \tag{17}$$

Optimization: It aims to select the k largest elements from $\mathbf{y}$:

$$\mathbf{z}^\star(\mathbf{y}) = \arg \mathrm{topk}(\mathbf{y}) . \tag{18}$$

Dataset: The item is randomly generated by uniform distribution on interval $[0, 1)$. We set up 50 items with a budget of 5. This dataset is generated from scratch and does not involve datasets by other entities.

C.5 Bipartite Matching

Graph matching in social networks has diverse applications, helping people find jobs and friends online. However, the edge relationships between nodes are sometimes unknown, necessitating the prediction of the edge connections prior to conducting matching. We formulate this problem following previous literature [72, 22, 44].

Prediction: Given the features $\mathbf{x}^i, \mathbf{x}^j$ of two nodes in each pair of nodes (i, j), the objective is to predict whether there exists an edge between these two nodes ($\mathbf{y}_{ij}$ is 1 if the link exists, otherwise 0).

$$\mathbf{y}^{ij} = \mathcal{M}([\mathbf{x}^i, \mathbf{x}^j]) . \tag{19}$$

Optimization: The optimization of matching formulates as a linear objective under a permutation constraint:

$$\mathbf{z}^\star(\mathbf{y}) = \arg \max_{\mathbf{z}} \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} \mathbf{y}^{ij} \mathbf{z}^{ij} \quad \text{s.t.} \quad \mathbf{z}\mathbf{1} = \mathbf{1} \quad \mathbf{z}^\top \mathbf{1} = \mathbf{1} . \tag{20}$$

where $\mathbf{y} \in \mathbb{R}^{N \times N}$ is the adjacency matrix and $\mathbf{z} \in \mathbb{R}^{N \times N}$ is the matching solution, and $\mathbf{z}^{ij} = 1$ indicates that node i matches j.

Dataset: We adopt the citation network Cora [48, 59, 76] [8] for the graph matching, where each node indicates a paper with a 1433-dimension feature obtained by bag-of-words methods. We split the full graph into 27 instances following the previous literature first proposed in [72] and later [22, 44], where each instance contains 100 nodes. Each instance constitutes a bipartite matching problem with cardinality 50.

License: The Cora dataset has been extensively utilized in numerous research studies in the domain of graph learning [39, 66]. According to our findings [9], it is licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) License.

C.6 Portfolio Optimization

The allocation of assets plays a pivotal role in facilitating the circulation of funds across various areas of society and promoting economic efficiency.

Prediction: We use historical features such as daily price and volume data to predict the return Y of the next day for N stocks.

Optimization: It aims to maximize the immediate return while reducing the risk with a classic Markowitz objective [47]:

$$\mathbf{z}^\star(\mathbf{y}) = \arg \max_{\mathbf{z}} \ \mathbf{z}^\top \mathbf{y} - \lambda \mathbf{z}^\top \mathbf{Q} \mathbf{z} \quad \text{s.t.} \quad \Sigma_{i=0}^{N} \mathbf{z}^i = 1 . \tag{21}$$

---

[8]http://www.cora.justresearch.com/
[9]https://www.openicpsr.org/openicpsr/project/100859/version/V1/view

20

where $0 \leqslant \mathbf{z}^i \leqslant 1$ is a continuous variable indicating the fraction of money invested in security i, $\lambda = 0.1$ is risk aversion constant, $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix representing the covariance between the returns of different securities.

Dataset: The data is obtained from public dataset S&P500 [56], the collection of 505 largest companies in the US market from 2004 to 2017. The features include previous returns of 10 days, weeks, months, and years, as well as rolling averages of these time windows. We set the risk aversion parameter $\alpha = 0.1$.

License: We obtained the publicly available dataset from the website [10], adhering to the following agreement [11] for its use.

C.7 Combinatorial Advertising for Inclusive Finance

As previously mentioned, the new dataset on Combinatorial Advertising comprises real industry advertising records, wherein a fintech platform collaborates with financial institutions to offer low-interest loans to users. The advertisers promote their financial services through mobile applications (APP) utilizing a combination of various channels, including in-app notifications, text messages, telephone calls, and other methods.

C.7.1 Predict-and-optimize formulation

The prediction is elaborated in the main paper, and we add optimization form as follows: **Optimization**: Denote $\mathbf{y}^{ij}$ the predicted conversion of user i on strategy (advertising combination) j, the optimization objective is:

$$
\begin{aligned}
\mathbf{z}^{\star}(\mathbf{y}) = \ &\arg\max_{\mathbf{z}} \ \Sigma_{i=1}^{N} \Sigma_{j=1}^{S} \mathbf{y}^{ij} \mathbf{z}^{ij} \\
\text{s.t.} \quad &\forall i, \ \Sigma_{j=1}^{S} \mathbf{z}^{ij} \leqslant 1 \\
&\Sigma_{j=1}^{S} \left(c^j \ \Sigma_{i=1}^{N} \mathbf{z}^{ij} \right) \leqslant B \ .
\end{aligned}
\tag{22}
$$

where $\mathbf{z}^{ij} \in \{0, 1\}$ is 1 if marketing strategy j is used for the user i. S is the total set of strategies and B is the total advertising budget. Note $|S| = 2^C$ where C is the number of advertising channels (app message, text message, etc.) and strategy j is the combination strategy of multiple channels. c^j is the cost for each strategy.

C.7.2 Data processing and model details

The data processing can be encapsulated as follows:

- **Data Collection**: The data was collected by an industrial company in an advertisement for financial loans across 2023 in 203 days on 269329 users.

- **Data Split**: The data has been split into distinct intervals, each spanning a period of 7 days. The dataset comprises a training set encompassing 161 days (comprising 23 instances) and a test set of 42 days (comprising 6 instances).

Within Each 7-Day Interval: (a) User Filtering: A filtering process has been undertaken to select users who have engaged with at least two distinct marketing channels. (b). Marketing Record Selection: For each marketing channel, the most recent marketing record is extracted and employed as input data. (c) Feature Extraction: The features, including encrypted user feature vector and marketing records, encompassing hours 1 to 6, are transformed into feature vectors comprising 41 distinct features.

- **Prediction**: A four-layer neural network architecture has been employed as the predictive model, where the hidden units are 128, 64,32, and 1, respectively. The model's input comprises user features, historical marketing records (push), and marketing combinations, while the output is a prediction of user conversion rates within combination j.

- **Optimization**: For the Integer Linear Programming formulation, we adopt OR-Tools solver [53] with SCIP (Strategic Consortium of Intelligence Professionals) [8] backend has been utilized for the resolution of the discrete linear programming problem. Specific parameter values have been assigned

---

[10]https://www.quandl.com/data/WIKI
[11]https://data.nasdaq.com/terms

21

to the optimization process, including channel costs (0.5 for channel 1, 1 for channel 2), costs for the combined strategy [0, 0.5, 1.0, 1.5] (which is simply treated as the sum of channel costs), and budget constraints, where the total budget is derived from the product of the number of users and the per-capital budget allocation of 0.1.

C.7.3 Evaluation

As mentioned above, the coefficient $\mathbf{y}$ is not possibly available since we cannot observe the response to different treatments of the same user at the same time. We refer to [25, 79] and introduce 'uplift' as a metric for the improvement of advertising brought by the certain treatment, defined as the difference of the value of treatment group $P^{\top}$ with control group P^{C}:

$$\text{Uplift} = P^{\top} - P^{C} = P(\mathbf{y} = 1 \mid \mathbf{w} = 1) - P(\mathbf{y} = 0 \mid \mathbf{w} = 0) \tag{23}$$

where w indicates the treatment and y indicates the label. In our case, the treatment group is the group of people whose combinatorial advertisement strategy is identical to the offline data, whereas the control group is the opposite.

C.7.4 License, Use Terms and Privacy for combinatorial advertising dataset

If you use the combinatorial advertising dataset provided in the benchmark, you agree to the following terms of use from "The Finvolution Group".

C.7.4.1 Data use terms and license

Acceptance of the Terms of Use

The Finvolution Group strives to enhance public access to and use of data that it collects and publishes, to promote data utilization and academic research. The data are organized in datasets (the"Datasets"). The Datasets are collections of data, managed by The Finvolution Group and provided in a number of machine-readable formats. The Finvolution Group provides you with access to the Datasets free of charge subject to the terms of this agreement (these "Dataset Terms"). Use of data derived from the Datasets, which may appear in formats such as tables and charts, is also subject to these Dataset Terms.

C.7.4.2 Licenses and Restrictions

You shall use the Datasets only for non-commercial research and educational purposes. You may provide research associates and colleagues with access to the Datasets provided that they first agree to be bound by these terms and conditions. If you are employed by a for-profit, commercial entity, your employer shall also be bound by these terms and conditions, and you hereby represents that he or she is fully authorized to enter into this agreement on behalf of such employer.

No Association

You may not use the name, any trade-mark, official mark, official emblem or logo of The Finvolution Group, or any of its other means of promotion or publicity, without The Finvolution Group's prior written consent nor in any event to represent or imply an association or affiliation with The Finvolution Group.

No Warranties

The Finvolution Group reserves the right at any time and from time to time to modify or discontinue, temporarily or permanently, this website, the Datasets, any means of accessing or utilizing the Datasets, or the API, at our sole discretion with or without prior notice to you.

The Finvolution Group may at our sole discretion, under any circumstances, for any or no reason whatsoever and with or without prior notice to you, terminate your access to the Datasets, any means of accessing or utilizing the Datasets or the API.

THE FINVOLUTION GROUP DISCLAIMS ALL WARRANTIES OF ANY KIND RELATED TO THE PROVISION OF THE DATASETS AND THE APIs.

Exclusion of Liability

22

THE FINVOLUTION GROUP SHALL NOT BE RESPONSIBLE OR LIABLE TO YOU FOR ANY LOSS OR DAMAGE OF ANY SORT INCURRED BY YOU IN CONNECTION WITH YOUR USE OF THE DATASETS. The Finvolution Group also shall not be responsible or liable for the accuracy, usefulness or availability of any data in the Datasets.

You acknowledge that these Dataset Terms constitute a non-exclusive agreement. The Finvolution Group may develop products or services that compete with products or services that you offer without incurring any liability.

Other parties may have ownership interests in some of the data and information ("Materials") contained on the Site. The Finvolution Group in no way represents or warrants that it owns or controls all rights in all Materials, and The Finvolution Group will not be liable to you for any claims brought against you by third parties in connection with your use of any Materials.

These Dataset Terms may be amended by The Finvolution Group from time to at our sole discretion. If there is a Chinese version of the Dataset Terms, the Chinese Dataset Terms shall prevail where the Chinese Dataset Terms conflict with the English Terms.

By continuing to use the Datasets subsequent to The Finvolution Group making available an amended version of these Dataset Terms, you acknowledge, agree and consent to such amendment.

No agency, partnership, joint venture, employee-employer, or franchiser-franchisee relationship is intended or created by these Terms and Conditions.

C.7.4.3 Privacy

Safeguarding user information and protecting user privacy is paramount in The Finvolution Group's operation since its inception. The Company has established a comprehensive administrative mechanism and standardized employee training system for stringent information security management. The Finvolution Group has also been deploying innovative technologies to promote user data protection. For example, the Company launched a Smart Finance Institute in 2018 for research and development in the field of artificial intelligence that can be applied in various aspects of financial services. In addition, The Finvolution Group is also a member of the National Information Security Standardization Technical Committee and Mobile Application (APP) Security Committee, maintaining up to date knowledge and compliant regarding the latest cyber-security regulatory requirements.

D Experimental Details

D.1 Analysis on Benchmark Results

On the knapsack (gen) and knapsack (energy) problems, SPO+, pointwise and listwise LTR, and LODL achieve lower regret than the two-stage method. For the energy scheduling problem, SPO+, LODL, and all statistical methods except pointwise-LTR exceed the two-stage approach. In the budget allocation problem, the LODL surprisingly does not outperform the two-stage approach, probably because it could be hard to learn a surrogate function for the non-linear objective. In the cubic top-K problem, optimal performance is achieved by the two-stage approach, which differs from the results of previous works [60] since a two-layer MLP is used as the predictive model. This may be attributed to the inherent simplicity of the sampled prediction data. The bipartite matching problem exhibits high regret on all models, probably due to the lack of samples in the train and test set, such that the over-smoothing issue may cause performance degradation. Most models have high regret for portfolio optimization, whereas pointwise-LTR and LODL have lower regret levels than the two-stage approach. In the combinatorial advertising problem, we do not run statistical class methods since the decision variables differ for processed instances, making the solution cache not applicable. For the surrogate class, the samples required to learn the surrogate objective function are hard to obtain. The suboptimal outcomes observed in continuous class methods may stem from the intricate nature of user behavior, posing challenges for the gradient backpropagation of convex functions.

D.2 Details of versatility of PnO in RQ3

This section explores sensitivity on the constraint (i.e. the capacity in knapsack problem) and variable size on the knapsack (gen) problem. We also show generalization results when the model is trained

23

| (a) Variable size of 40 | (b) Variable size of 60 | (c) Variable size of 80 |

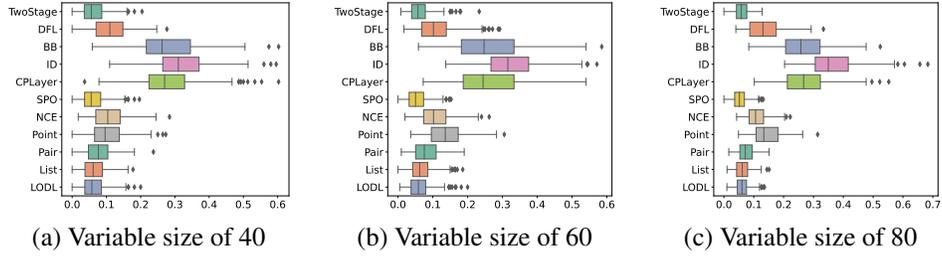



Figure 8: Regret on knapsack (gen) problem with increasing decision variable size (40, 60, 80). With an increasing decision variable number, performance fluctuates and degrades in some models (BB, ID, and pointwise-LTR).

| (a) Generalize to capacity 60 | (b) Generalize to capacity 90 | (c) Generalize to capacity 120 |

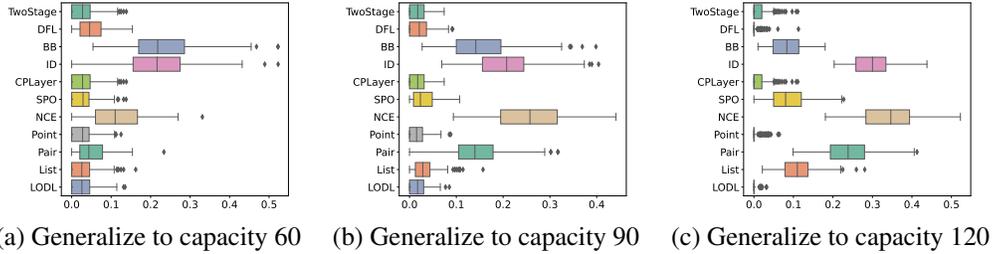



Figure 9: Regret on knapsack (gen) problem when generalizing trained model on capacity 30 to 60, 90 and 120. Most end-to-end trained models degrade in the generalization task.

Table 8: Average train/test time per epoch (in second) with variable size 40, 60, 80, 100, 200 on Knapsack (Gen) Dataset. Note that LODL data generation time is not listed but gradually escalates, becoming the primary bottleneck.

| | Setting | Predictive | Discrete | | | Continuous | | Statistical | | | | Surrogate |
		Two-stage	DFL	Blackbox	Identity	CPLayer	SPO+	NCE	point-LTR	pair-LTR	list-LTR	LODL
Size 40	Train Time	0.061	1.634	1.628	2.121	0.671	1.376	2.516	5.566	17.954	4.490	4.818
	Test Time	1.725	2.141	2.888	7.078	2.317	3.450	4.228	5.404	30.909	8.122	1.619
Size 60	Train Time	0.078	2.397	1.737	12.709	0.575	1.585	3.328	10.530	24.310	4.888	6.562
	Test Time	2.111	2.878	2.439	44.366	4.192	4.188	8.221	10.787	51.412	11.987	2.063
Size 80	Train Time	0.882	5.839	2.299	27.327	0.649	3.170	4.455	19.337	116.366	8.780	9.190
	Test Time	3.594	4.703	3.969	108.740	4.676	7.577	16.063	18.999	69.713	21.053	2.295
Size 100	Train Time	1.025	3.197	5.637	2.884	0.636	4.158	4.536	21.758	114.583	25.009	11.318
	Test Time	4.355	3.968	8.579	6.486	4.617	10.474	14.705	20.370	493.726	47.784	2.816

on the optimization problem of capacity 30 and tested directly on capacities 60 and 90. The detailed results for sensitivity analysis Table 7.

In Figure 8, we evaluate the knapsack problem by gradually increasing the number of variables (with constraints proportionally scaling accordingly) to observe changes in the model. The running time is also reported in Table 8. We set the decision variable size of 40, 60, and 80 and observe that certain models, such as BB, ID, and point-wise LTR, exhibit fluctuations and degradations to different extents. Some models demonstrated more stable performance, including the two-stage and statistical and surrogate approaches, which could be attributed to the reliance on the relative importance of multiple solutions, which is less sensitive to the change of decision variable numbers.

Lastly, in Figure 9, we conduct experiments to investigate the generalization performance by deploying models trained on the dataset of capacity 30 directly to problems with constraints of 60, 90, and 120. The figure indicates that certain models, including the two-stage model, LODL, and pointwise LTR, demonstrated the ability to prevent performance deterioration. Some models even exhibited improvements, such as the continuous-class models cvxpy layer and SPO. Conversely, several models experienced a decline in performance, notably the discrete-class Blackbox and Identity models, as well as statistical-class models like NCE and LTR.

24

Table 9: Sensitivity analysis on hyper-parameters (hidden units, number of training epochs) on Knapsack (Energy) evaluated by regret(Best three: red, orange, blue.

Setting		Predictive Two-stage	Discrete DFL	Blackbox	Identity	Continuous CPLayer	SPO+	Statistical NCE	point-LTR	pair-LTR	list-LTR	Surrogate LODL
Hidden units	8	9.456	9.467	20.218	21.352	20.430	**7.824**	11.778	8.606	8.534	7.904	9.339
	32	8.745	8.353	35.705	17.156	36.402	8.407	11.932	8.236	9.022	**8.083**	9.567
	128	8.319	8.318	45.809	45.149	45.948	8.979	12.028	8.317	8.758	7.794	9.127
Train epochs	100	9.178	9.108	35.705	22.013	36.402	8.407	11.932	8.236	9.840	**8.083**	9.567
	300	8.745	8.353	35.705	17.156	36.402	8.407	11.932	8.236	9.022	**8.083**	9.567
	500	8.745	8.353	35.705	16.826	36.402	8.407	11.932	8.236	9.840	**8.083**	9.567

Table 10: Sensitivity analysis on the number of training samples) on Knapsack (Gen) evaluated by regret(Best three: red, orange, blue.

Setting		Predictive Two-stage	Discrete DFL	Blackbox	Identity	Continuous CPLayer	SPO+	Statistical NCE	point-LTR	pair-LTR	list-LTR	Surrogate LODL
Num of samples	400	6.595	11.744	24.274	31.874	24.769	6.223	13.438	6.402	7.820	**6.031**	6.044
	2000	6.389	25.017	6.595	6.595	24.480	6.168	19.207	**5.590**	8.385	6.471	-

Table 11: Theoretical assumptions and insights of PnO in literature. "LR", "RF" and "KNN" is short for linear regression, random forest, and K-nearest-neighbor.

Paper	Prediction model	Optimization form	Perfect prediction assumption	Theoretical insight
[4]	Not Specified	Submodular	Required	Exists submodular functions that no algorithm can obtain an approximationstrictly better than $1/2 - O(1)$ using polynomially-many samples.
[13]	Not specified	Submodular	Required	With structure information, achieve a constant approximation for the maximum coverage problem
[7]	LR, RF, KNN, etc.	All	Required	Under appropriate conditions and for certain predictive prescriptions, prescription converge to true full information optimizers
[9]	All	All	Not required	PnO is better than PtO by making the right error trade-offs

D.3 Sensitivity analysis of training on hyperparameters

This section explores the sensitivity of each model on hyper-parameters during training, as shown in Table 9. To begin with, we explore the influence of hidden size on the ultimate decision. We observe that with increasing hidden size, the two-stage method's result improves slightly. Listwise-LTR remains a good performance at different hidden sizes, but SPO+ achieved the best score at the unit of 8, maybe because a larger neural network would cause SPO+ to overfit. Then, we show the sensitivity results with respect to the number of training epochs. We observe that the two-stage method improves slightly from 100 to 300, while most methods have converged at the epoch of 300. Last, we explore the sensitivity analysis with respect to the number of samples (400 and 2000 in the training set) on the knapsack (gen) dataset, shown in Table 10. We observe that the two-stage method improves slightly, and SPO+ achieves the best results with more training samples. The performance of some PnO methods degrades slightly, probably due to instability by more samples. Note the result of LODL is not applicable since the data generation time exceeds the threshold.

D.4 Theoretical Interpretation on benchmarks

Though this work is focused on benchmarking PnO problems, we also attempt to discuss the literature on PnO from a theoretical point of view. As shown in Table 11, previous literature [4] discusses the hardness of solving submodular function when the optimization function is unknown, and [13] proposes that under certain conditions (where structure information is given), a constant-approximation algorithm could be designed. In the context of bipartite matching problems within our framework, a correlation exists wherein the prediction of graph structure precedes the determination of matching. This could be helpful to this benchmark given the observed high decision regret.

Then, the recent work [7] also discusses the capability of PtO. However, it is built on a strong assumption that the prediction is perfect without error, which makes it hard to be applicable to real scenarios. In the real world and in this work, we are more focused on the "perfect prediction" of some

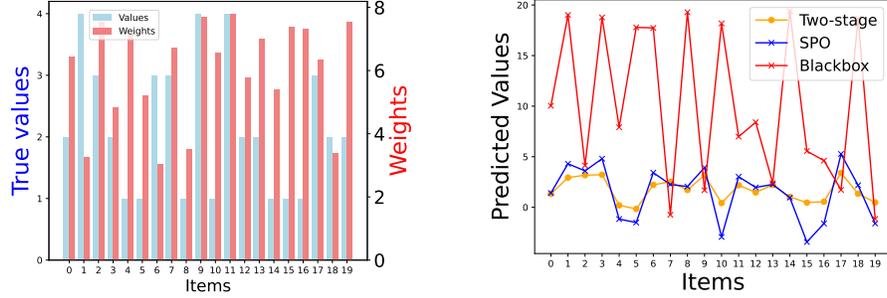
25

(a) Values and weights of knapsack problem   (b) Predicted values for various methods

Figure 10: Qualitative analysis for knapsack problem on the 10-th test instance.

distribution on the dataset and investigate how to propose and deploy end-to-end models that achieve better ultimate decisions when the prediction is not perfect. [9] proposes an insightful perspective that end-to-end training of PnO is better than the two-stage training of PtO even when the prediction is perfect, since PnO models make the right error trade-off. This perspective could also be validated partially by the experiment shown in Figure 5(c d), where the prediction curve and decision curve perform differently by some models (e.g., Blackbox).

D.5   Qualitative Analysis

In Fig 5, we visually scale the prediction loss of Blackbox to $5 * 10^9$ for clarity, while others to 10^8.

We conduct a set of qualitative analyses on the knapsack problem to explore the reasons behind the success or failure of these methods. First, Fig 10(a) visualizes a bar plot of the 10-th test instance in the knapsack (gen) problem instance, showing the values and weights. We can see that the range of predicted values by SPO is roughly the same as that of the two-stage model, but the Blackbox model, without labels, has a significant difference from the true values. In Fig 10 (b) we plot the predicted values for 20 items for three models: the two-stage model, the well-performing SPO model, and the poorly-performing Blackbox model.

We can observe that the SPO model achieved the best decision quality (with the objective value of 18, selecting items 1, 3, 6, 9, 17, and 18). The two-stage model's decision quality was slightly lower (with the objective value of 17, selecting items 1, 3, 6, 8, 9, and 17). SPO improved decision quality by replacing item 9 with item 18, by predicting a higher predicted value to item 18, and by lowering the prediction for item 9, which is guided by information of optimization objective while the two-stage does not involve this. Although this increased the prediction error, it enhanced the decision quality.

For the Blackbox model, we observed that its predicted values were similar to SPO for some items, but due to the lack of label supervision during the prediction phase (not using coefficient labels), the prediction errors were larger, leading to poorer decision quality.

E   Terms of Use, Limitations and Broader Impacts

Terms of Use As part of our benchmark work, we utilized numerous datasets in the benchmarking process, and we adhere to the respective usage terms and licenses elaborated in Appendix C. Additionally, we are releasing a new industrial dataset, and the license, use terms, and privacy are also elaborated in Appendix C.7.4. If readers use our benchmark dataset, please also adhere to the corresponding user terms specified by them.

Limitations The limitations of this paper include the inherent difficulty in analyzing the internal mechanisms of PnO. Aside from a few existing works that examine the theoretical underpinnings of PnO (like SPO [20]), at present, it is challenging to provide theoretical explanations or straightforward criteria to determine when PnO outperforms PtO or which specific model is the best choice. Therefore, our focus for now is primarily on the empirical results of these methods on current datasets. To assist practitioners, we have categorized and analyzed these methods based on their generalizability, usage

26

conditions, and associated costs, and our experimental results. In Appendix D.4, we also attempt to interpret the benchmark results based on existing theories for the reader's reference.We offer a predict-and-optimize model using deployment recommendations in Sec. 3.5 based on these above considerations for practitioners.

Our work does not aim to cover all combinatorial optimization problems, predictors, solvers, or Predict-and-Optimize (PnO) methods. Instead, we present a user-friendly framework with representative implementations for each module. Readers who wish to implement their deployments in any module can do so by replacing the corresponding block within our framework.

Broader Impacts As far as we are concerned, this paper does not have critical negative societal impacts. We hope that this benchmark work can support better decision-making under uncertainty by utilizing end-to-end prediction and optimization methods for training predictive models more appropriately. In Appendix C, we also discuss the possible positive implications of these decisions, including more energy-efficient production scheduling models, more rational resource allocation under uncertain scenarios, more appropriate social matching, more interpretable models, more appropriate capital allocation, and more inclusive financial services. Certainly, we acknowledge that similar to previous works on predict-and-optimize (PnO), decision-making under uncertainty could potentially be applied to inappropriate purposes. However, given the current challenges of deploying PnO in industry and practice, we consider the negative impacts will be minimal. We hope that this benchmark can facilitate future advances in this field and aid in creating better decisions under uncertain scenarios for people.

27

Checklist

1. For all authors...

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The main claims in the abstract and introduction accurately reflects the paper's contributions and scope.

    (b) Did you describe the limitations of your work? [Yes] We discuss the limitations in Appendix E.

    (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss the impacts in Appendix E.

    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We read the ethics review guidelines and ensure the paper conforms to them.

2. If you are including theoretical results...

    (a) Did you state the full set of assumptions of all theoretical results? [N/A] We make some attempts to explain the benchmark results with existing theories in Appendix D.4, but it is not our primary focus.

    (b) Did you include complete proofs of all theoretical results? [N/A] We make some attempts to explain the benchmark results with existing theories in Appendix D.4, but it is not our primary focus.

3. If you ran experiments (e.g. for benchmarks)...

    (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] We specified the model details in Sec. 4.3 and Appendix B. The code will be open-sourced after publication.

    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify training details in Appendix C and D

    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for knapsack problem in Fig 7.

    (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] We specify the used computing resource in Sec 4.3.

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] We cite the dataset sources in Table 2, and specified all the dataset details in Appendix C.

    (b) Did you mention the license of the assets? [Yes] We discuss the license of assets in Appendix C.

    (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We introduce our new dataset in Appendix C.7.

    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We discuss in Appendix C.7.4.

    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We discuss in Appendix C.7.4.

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] The paper does not use crowdsourcing or conducted research with human subjects.

    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] The paper does not use crowdsourcing or conducted research with human subjects.

    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] The paper does not use crowdsourcing or conducted research with human subjects.

28