Goal-Conditioned Supervised Learning for Multi-Objective Recommendation

Shijun Li[1], **Hilaf Hasson**[2], **Jing Hu**[3], **Joydeep Ghosh**[4]
[1,4]The University of Texas at Austin, [2]Intuit AI Research, [3]Intuit Inc.
{shijunli, jghosh}@utexas.edu, {hilafhasson, hu.jing000}@gmail.com

Abstract

Multi-objective learning endeavors to concurrently optimize multiple objectives using a single model, aiming to achieve high and balanced performance across diverse objectives. However, this often entails a complex optimization problem to balance the learning of potentially conflicting objectives, leading to solutions with higher memory requirements and computational complexity. This paper introduces a Multi-Objective Goal-Conditioned Supervised Learning (MOGCSL) framework for automatically learning to achieve multiple objectives from offline sequential data. MOGCSL extends the conventional GCSL method to multi-objective scenarios by redefining goals from one-dimensional scalars to multi-dimensional vectors. It benefits from naturally eliminating the need for complex architectures and optimization constraints. Moreover, MOGCSL inherently disentangles uninformative or noisy training instances that fail to achieve desirable long-term rewards across multiple objectives. We also introduce a novel goal-selection algorithm for MOGCSL to model and identify desired and achievable goals for inference.

In this paper, we focus on its application to the next action prediction problem in commercial-grade recommender systems. In this context, any viable solution needs to be reasonably scalable and also be robust to large amounts of noisy data that is characteristic of this application space. We show that MOGCSL performs admirably on both counts by extensive experiments. Also, analysis and experiments are included to explain its strength in discounting the noisier portions of training data in recommender systems with multiple objectives. Code is available at: https://github.com/xiwenchao/MOGCSL.

1 Introduction

Multi-objective learning techniques typically aim to train a single model to determine a policy for multiple objectives, which are often defined on different types of tasks [1–3]. For instance, two common tasks in recommender systems are to recommend items that users may click or purchase, and hence the corresponding two objectives are defined as pursuing higher click rate and higher purchase rate. However, learning for multiple objectives simultaneously is often nontrivial, particularly when there are potential inherent conflicts among these objectives [2].

Existing approaches to multi-objective learning broadly address this optimization issue by formulating and optimizing a loss function that takes into account multiple objectives in a supervised learning paradigm. Some research focused on designing model architectures specifically for multi-objective scenarios, such as the shared-bottom structure [4], the experts and task gating networks [4], and so on. Another line of research studies how to constrain the optimization process based on various assumptions regarding how to assign reasonable loss weights [5] or dynamically adjust gradients [6]. We remark that these approaches often introduce substantial space and computational complexity

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

[4, 3]. Also, they treat all the data samples uniformly. In recommender systems, sometimes none of the items shown to a user is of interest, rendering their choice uninformative. Also users are often distracted or their interests temporarily change. These are some of the reasons that the observed interaction data in real settings suffers from substantial uninformative or noisy components that are better left discounted. Existing approaches do not cater well to this aspect of our focus application.

To address these issues, we propose a novel method called **Multi-Objective Goal-Conditioned Supervised Learning (MOGCSL)**. In this framework, we first apply goal-conditioned supervised learning (GCSL) [7] to the multi-objective recommendation problem by introducing a multi-dimensional goal. At its core, MOGCSL aims to learn primarily from the behaviors of those sessions where the long-term reward ends up being high, thereby discounting noisy user choices coming with low long-term rewards. Unlike conventional GCSL, however, we represent the reward gained from the environment with a vector instead of a scalar. Each dimension of this vector indicates the reward for a certain objective. The "goal" in MOGCSL can then be defined as a vector of desirable cumulative rewards on all of the objectives given the current state. By incorporating these goals as input, MOGCSL learns to rely primarily on high-fidelity portions of the data with less noise corresponding to multiple objectives, which helps to better predict users' real preference. Extensive experiments indicate that MOGCSL significantly outperforms other baselines and benefits from lower complexity.

For inference of GCSL, most previous works employ a simple goal-choosing strategy for scalar goals [8, 9] (e.g., some multiple of the average goal in training). To study how to choose reasonable and desirable goals within multi-objective scenarios, we introduce a goal-choosing algorithm that estimates the distribution of achievable goals using variational auto-encoders, and automates the selection of desirable high goals on multiple objectives as input for inference. This algorithm addresses the key challenge of selecting achievable "high" multi-dimensional goals, a problem that has remained unexplored yet. By comparing the proposed method with simpler statistical strategies, we gain valuable insights into the goal-choosing process and trade-offs for practical implementation.

Our key contributions can be summarized as follows:

- We introduce a supervised framework called MOGCSL for multi-objective recommendations that, by design, selectively leverages training data with desirable long-term rewards on multiple objectives for inference. We implement this approach using a transformer encoder optimized with a standard cross-entropy loss, avoiding more complex architectures or optimization constraints that are typical for multi-objective learning. Empirical experiments on real-world e-commerce datasets demonstrate the superior efficacy and efficiency of MOGCSL.
- As a part of MOGCSL, we conduct a formal analysis of goal properties for inference with a theorem. Then we introduce a goal-choosing algorithm that can model the distribution of achievable goals over interaction sequences and choose desirable "high" goals across multiple objectives.
- We conduct a comprehensive analysis of MOGCSL's working mechanism and are the first to reveal its ability to effectively mitigate harmful effects of noisy instances in the training data, which is a crucial challenge in practical applications like recommender systems.

2 Related Work

Multi-Objective Learning. Multi-objective learning typically investigates the construction and optimization of models that can simultaneously achieve multiple objectives. Existing research focuses mainly on resolving the problem by model architecture designs [4, 10] and optimization constraints [5, 11]. All of these works give equal weight to all instances in the training data that they learn from, instead of forcefully distinguishing noisy data from non-noisy data by considering their different effects on multiple objectives. This is fine in many applications but problematic in commercial recommendation systems. Moreover, these approaches often introduce substantial space and computational complexity [3], making them more challenging for large-scale applications in the real world. For example, MMOE [4] requires constructing separate towers for each objective. DWA [5] necessitates recording and calculating loss change dynamics for each training epoch, while PE [11] demands substantial computational resources to solve an optimization problem for Pareto efficiency. A recent work FAMO [12] aims to improve the efficiency of multi-objective learning, but still suffers from the complexity of calculating gradient similarities to determine task weights.

Goal-Conditioned Supervised Learning. In contrast, we propose resolving the multi-objective optimization dilemma within the framework of GCSL [7, 8, 13]. Typically, GCSL can be directly combined with various sequential models with minor adaptations and trained entirely on offline data. This paradigm effectively transforms offline reinforcement learning into a supervised learning problem. However, as far as we know, most existing works focus on optimizing a single objective. Our work extends GCSL to the multi-objective setting, eliminating the need for scalarization functions or other constraints during training. Additionally, although some works have explored how to assign more valuable goals to enhance GCSL training [14, 15], the properties and determination inference goals, especially multi-dimensional ones, remain less explored. We propose a novel algorithm to model the achievable goals and automatically choose desirable goals on multiple objectives as input during inference stage. Note that we are solving for next action prediction problem, and use long-term rewards only as extra information, unlike multi-objective reinforcement learning approaches that aim to maximize cumulative returns across multiple objectives [16, 17]. Also, their evaluation principles are different, typically relying on long-term metrics and synthetic environments.

3 Methodology

In this section, we first illustrate the general optimization paradigm of MOGCSL. Then we expound on the training process of MOGCSL and the proposed goal-choosing algorithm for inference. Furthermore, we give a detailed analysis of the denoising capability of MOGCSL.

3.1 A New View for Multi-Objective Learning

Multi-objective learning is typically formulated as an optimization problem over multiple losses [4, 10, 6, 11], each defined on a distinct objective. Consider a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i^1, y_i^2, ..., y_i^n)\}$, where $\mathbf{x}_i$ represents the feature and y_i^j is the ground-truth score on the j-th objective. For a given model $f(\mathbf{x}; \boldsymbol{\theta})$, multiple empirical losses can be computed, one per objective as $\mathcal{L}_j(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y^j)\in\mathcal{D}}[\mathcal{L}(f(\boldsymbol{x};\boldsymbol{\theta}), y^j)]$. The model can then be optimized by minimizing a single loss, which is obtained by combining all the losses through a weighted summation as: $\min_{\boldsymbol{\theta}} \sum_{j=1}^{n} w^j \mathcal{L}_j(\boldsymbol{\theta})$.

A fundamental question is how to assign these weights and how to regulate the learning process to do well on all the objectives concurrently. Earlier research sought to address this issue based on assumptions regarding the efficacy of certain model architectures or optimization constraints, which may not be generally valid and can significantly increase complexity [3].

In contrast, we propose to approach the learning and optimization for multi-objective learning from a different perspective. Specifically, we posit that the interaction process between the agent and the environment can be formalized as a Multi-Objective Markov Decision Process (MOMDP) [18]. Denote the interaction trajectories collected by an existing agent as $\mathcal{D} = \{\tau_i\}_{i\in[1,M]}$. In the context of recommender systems, each trajectory τ records a complete recommendation session between a user entering and exiting the recommender system, such that $\tau = \{(\boldsymbol{s}_t, a_t, \boldsymbol{r}_t)\}_{t\in[1,|\tau|]}$. A state $\boldsymbol{s}_t \in \mathcal{S}$ is taken as the representation of user's preferences at a given timestep t. An action a_t is recommended from the action space $\mathcal{A}$ which includes all candidate items, such that $|\mathcal{A}| = |\mathcal{V}| = N$ where $\mathcal{V}$ denotes the set of all items. $R(s_t, a_t)$ is the reward function, where $\boldsymbol{r}_t = R(s_t, a_t)$ means the agent receives a reward $\boldsymbol{r}_t$ after taking an action a_t under state $\boldsymbol{s}_t$. Note that the reward function $R(\boldsymbol{s}_t, a_t)$ in MOMDP is represented by a multi-dimensional vector instead of a scalar.

In this context, all the objectives can be quantified using reward $\boldsymbol{r}_t$ at each time step. Specifically, since $\boldsymbol{r}_t$ is determined by user's behavior in response to recommended items, it naturally reflects the recommender's performance on these objectives. For example, if the user clicks the recommended item a_t, the value on the corresponding dimension of $\boldsymbol{r}_t$ can be set to 1; otherwise, it remains 0. In sequential recommendation scenarios, the target of the agent is to pursue better performance at the session level. Session-level performance can be evaluated by the cumulative reward as:

$$\boldsymbol{g}_t = \sum_{t'=t}^{|\tau|} \boldsymbol{r}_{t'}, \tag{1}$$

where $\boldsymbol{g}_t$ can be called as a "goal" in the literature of goal-conditioned supervised learning [19].

Then, the target of multi-objective learning for recommender systems can be formulated as *developing a policy that achieves satisfactory performance (i.e., the goals) across multiple objectives in*

Algorithm 1: Training of MOGCSL

Input: training data $\mathcal{D}_{tr}$, batch size B, model parameters θ
Intialization: initialize parameters θ
Relabel all the rewards with goals according to Eq. (1)
repeat
  Randomly sample a batch of $(\boldsymbol{s}_t, a_t, \boldsymbol{g}_t)$ from $\mathcal{D}_{tr}$
  Compute the representation $\boldsymbol{M}_{\boldsymbol{s}_t,\boldsymbol{g}_t}$ via Eq. (2)-Eq.(3)
  Derive the prediction logits via Eq. (4)
  Calculate the loss function $\mathcal{L}(\theta)$ via Eq. (5)
  Update θ by minimizing $\mathcal{L}(\theta)$ with stochastic gradient descent: $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$
until *convergence*

recommendation sessions. In this research, we address this problem within the framework of GCSL. During the training stage, the aim is to determine the optimal action to take from a given current state in order to achieve the specified goal. The agent, denoted as π_θ, is trained by maximizing the likelihood of trajectories in the offline training dataset $\mathcal{D}_{tr}$ through an autoregressive approach, expressed as $\arg\max_\theta \mathbb{E}_{\mathcal{D}_{tr}}[log\pi_\theta(a|\boldsymbol{s}, \boldsymbol{g})]$. Notably, there are no predefined constraints or assumptions governing the learning process. During the inference stage, when an achievable and desirable goal is specified, the model is expected to select an action based on the goal and the current state, with the aim of inducing behaviours to achieve that goal.

3.2 MOGCSL Training

The initial step of MOGCSL training is relabeling the training data by substituting the rewards with goals. Specifically, for each trajectory $\tau \in \mathcal{D}_{tr}$, we replace every tuple $(\boldsymbol{s}_t, a_t, \boldsymbol{r}_t)$ with $(\boldsymbol{s}_t, a_t, \boldsymbol{g}_t)$, where $\boldsymbol{g}_t$ is defined according to Eq. (1). Subsequently, we employ a sequential model [20] based on Transformer-encoder (denoted as *T-enc*) to encode the users' sequential behaviors and obtain state representations. We chose a transformer-based encoder due to its widely demonstrated capability in sequential recommendation scenarios [20, 21]. Specifically, let the interaction history of a user up to time t be denoted as $v_{1:t-1} = \{v_1, ..., v_{t-1}\}$. We first map each item $v \in \mathcal{V}$ into the embedding space, resulting in the embedding representation of the history: $\boldsymbol{e}_{1:t-1} = [\boldsymbol{e}_1, ..., \boldsymbol{e}_{t-1}]$. Then we encode $\boldsymbol{e}_{1:t-1}$ by *T-enc*. Since the current timestep t is also valuable for estimating user's sequential behavior, we incorporate it via a timestep embedding denoted as emb_t through a straightforward embedding table lookup operation. Similarly, we derive the embedding of the goal $emb_{\boldsymbol{g}_t}$ through a simple fully connected (FC) layer. The final representation of state $\boldsymbol{s}_t$ is derived by concatenating the sequential encoding, timestep embedding and goal embedding together:

$$emb_{\boldsymbol{s}_t} = Concat(\textit{T-enc}(\boldsymbol{e}_{1:t-1}), emb_t, emb_{\boldsymbol{g}_t}). \tag{2}$$

To better capture the mutual information, we feed the state embedding into a self-attention block:

$$\boldsymbol{M}_{\boldsymbol{s}_t,\boldsymbol{g}_t} = Atten(emb_{\boldsymbol{s}_t}). \tag{3}$$

Then we use an MLP to map the derived embedding into the action space, where each logit represents the preference of taking a specific action (i.e., recommending an item):

$$[\pi_\theta(v^1|\boldsymbol{s}_t, \boldsymbol{g}_t), ..., \pi_\theta(v^N|\boldsymbol{s}_t, \boldsymbol{g}_t)] = \delta(MLP(\boldsymbol{M}_{\boldsymbol{s}_t,\boldsymbol{g}_t})), \tag{4}$$

where v^i denotes the i-th item in the candidate pool, δ is the soft-max function, and θ denotes all parameters of this model. The model structure is shown in Appendix 1.

The training objective is to correctly predict the subsequent action that is mostly likely lead to a specific goal given the current state. As discussed in Section 3.1, each trajectory of user's interaction history represents a successful demonstration of reaching the goal that it actually achieved. As a result, the model can be naturally optimized by minimizing the expected cross-entropy as:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\boldsymbol{s}_t,a_t,\boldsymbol{g}_t)\in\mathcal{D}_{tr}}[-log(\pi_\theta(a_t|\boldsymbol{s}_t, \boldsymbol{g}_t))]. \tag{5}$$

3.3 MOGCSL Inference

After training, we derive a model $\pi_\theta(a|\boldsymbol{s}, \boldsymbol{g})$ that predicts the next action based on the given state and goal. However, while the goal can be accurately computed through each trajectory in the training data

Algorithm 2: Inference of MOGCSL

Input: state $\boldsymbol{s}'$, sample size K, policy model π, utility principle U, prior $q(\boldsymbol{g}'|\boldsymbol{s}')$, distribution of achievable goals $P(\boldsymbol{g}^a|\boldsymbol{s}', \boldsymbol{g}', \pi)$
Intialization: set of potential input goals $\boldsymbol{G}' = \emptyset$, set of expected achievable goals $\boldsymbol{G}^a = \emptyset$
for $k = 1, \ldots, K$ **do**
  Sample a $\boldsymbol{g}'_k$ from $q(\boldsymbol{g}'|\boldsymbol{s}')$
  Compute the expectation of the achievable goal through sampling: $\tilde{\boldsymbol{g}}^a_k = \mathbb{E}_{\boldsymbol{g}^a_k \sim P(\cdot|\boldsymbol{s}', \boldsymbol{g}'_k, \pi)} \boldsymbol{g}^a_k$
  $\boldsymbol{G}' = \boldsymbol{G}' \cup \boldsymbol{g}'_k$
  $\boldsymbol{G}^a = \boldsymbol{G}^a \cup \tilde{\boldsymbol{g}}^a_k$
Choose the best $\tilde{\boldsymbol{g}}^a_b$ from $\boldsymbol{G}^a$ according to $U(\tilde{\boldsymbol{g}}^a)$
Choose corresponding $\boldsymbol{g}'_b$ from $\boldsymbol{G}'$
Return: $\pi(\cdot|\boldsymbol{s}', \boldsymbol{g}'_b)$

via Eq. (1), we must assign a desirable goal as input for the new state encountered during inference. Classic GCSL approaches typically determine this goal-choosing strategy based on simple statistics calculated from the training data. E.g., Chen et al. [8] and Zheng et al. [22] set the goals for all states at inference as the product of the maximal cumulative reward in training data and a fixed factor serving as a hyperparameter. Similarly, Xin et al. [9] derive the goals for inference at a given timestep by scaling the mean of the cumulative reward in training data at the same timestep with a pre-defined factor. However, a central yet unexplored question is: *what are the general characteristics of the goals and how can we determine them for inference in a principled manner?*

In this paper, we investigate the distribution of the multi-dimensional goals that can be achieved during inference by first stating the following theorem. Proof is given in Appendix A.

Theorem 1. *Assume that the environment is modeled as an MOMDP. Consider a trajectory τ that is generated by the policy $\pi(a|\boldsymbol{s}, \boldsymbol{g})$ given the initial state $\boldsymbol{s}_1$ and goal $\boldsymbol{g}_1$, the distribution of goals $\boldsymbol{g}^a$ that the agent actually achieves throughout the trajectory is determined by $(\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$.*

Based on this theorem, we'd like to learn the distribution of $\boldsymbol{g}^a$ conditioned on $(\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$, denoted as $P(\boldsymbol{g}^a|\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$. General generative models, such as GANs [23] and diffusion models [24], can be employed to learn this distribution. In this paper, we propose the use of a conditional variational auto-encoder (CVAE) [25] due to its simplicity, robustness, and ease of formulation.

Specifically, this distribution can be learned directly on the training data $\mathcal{D}_{tr}$. For each $(\boldsymbol{s}, \boldsymbol{g}) \in \mathcal{D}_{tr}$, $\boldsymbol{g}$ should be a sample from the distribution of the achievable goals by the policy π, given the initial state $\boldsymbol{s}$ and input goal $\boldsymbol{g}$. That's because the policy π is trained to imitate the actions demonstrated by each data point in $\mathcal{D}_{tr}$, where the achieved goal of the trajectory starting from $(\boldsymbol{s}, \boldsymbol{g})$ is exactly $\boldsymbol{g}$. Let $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{g}, \pi)$. The loss function is:

$$\mathcal{L}_{CVAE1} = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{g}) \in \mathcal{D}_{tr}, z \sim Q_1}[-logP_1(\boldsymbol{g}|z, \boldsymbol{c}) + D_{KL}(Q_1(z|\boldsymbol{g}, \boldsymbol{c})||P(z))], \tag{6}$$

where $Q_1(z|\boldsymbol{g}, \boldsymbol{c})$ is the encoder and $P_1(\boldsymbol{g}|z, \boldsymbol{c})$ is the decoder. Based on Gaussian distribution assumption, they can be written as $Q_1 = \mathcal{N}(\mu(\boldsymbol{g}, \boldsymbol{c}), \Sigma(\boldsymbol{g}, \boldsymbol{c}))$ and $P_1 = \mathcal{N}(f_{CVAE1}(z, \boldsymbol{c}), \sigma^2 I)$ respectively, where $z \sim \mathcal{N}(0, I)$. Then we can derive a sample of $\boldsymbol{g}^a$ by inputting a sampled z into f_{CVAE1}. We include more discussions of this sampling-based strategy in Appendix C.6.

On the inference stage, given a new state $\boldsymbol{s}'$, we first sample a set of goals $\boldsymbol{g}'$ as the possible input of π through a learnable prior $q(\boldsymbol{g}'|\boldsymbol{s}')$. Similarly, we learn this prior via another CVAE on the training data. The loss is:

$$\mathcal{L}_{CVAE2} = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{g}) \in \mathcal{D}_{tr}, z \sim Q_2}[-logP_2(\boldsymbol{g}|z, \boldsymbol{s}) + D_{KL}(Q_2(z|\boldsymbol{g}, \boldsymbol{s})||P(z))]. \tag{7}$$

Finally, we'll choose a desirable goal as the input along with the new state $\boldsymbol{s}'$ encountered in inference by sampling from the two CVAE models. Specifically, we propose to: (1) sample from the prior $q(\boldsymbol{g}'|\boldsymbol{s}')$ to get a set of potential input goals, denoted as $\boldsymbol{G}'$, (2) for each $\boldsymbol{g}' \in \boldsymbol{G}'$, estimate the expectation of the actually achievable goal $\tilde{\boldsymbol{g}}^a$ by sampling from $P(\cdot|\boldsymbol{s}', \boldsymbol{g}', \pi)$ and taking the average , (3) choose a best goal as input for inference from $\boldsymbol{G}'$ according to the associated expected $\tilde{\boldsymbol{g}}^a$ by a predefined utility principle $U(\tilde{\boldsymbol{g}}^a)$, which generally tends to pick up a "high" goal to achieve larger rewards on multiple objectives. The exact definition can be flexible with practical requirements with

different objective importance. E.g., choose by a predefined partial ordering. Note that our algorithm is general and can accommodate adaptations of the utility function, as the desired goals are always set among the set of achievable goals, which naturally resolves the objective-conflicting problem. This definition flexibility allows practitioners to easily tailor MOGCSL to various practical scenarios and objective priorities. See Algorithm 2 for inference pseudocode.

3.4 Analysis of Denoising Capability

An important benefit of MOGCSL is its capability to remove harmful effects of potentially noisy instances in the training data by leveraging the multiple-objective goals. To illustrate this, we consider the following common setup in recommender systems. There is a recommender system that has been operational, and recording data. At each interaction, the system shows the user a list of items. The user then chooses one of these items. In the counterfactual that the recommender system is ideal, the action recorded would be a which reveals the user's true interest. Since the actual recommender system to collect the data is not ideal, we have no direct access to a, but rather to a noisy version $\varepsilon(a)$.

We assume that the noisy portion of the training data originates from users who are presented with a list of items that are not suitable for them, rendering their reactions to these recommendations uninformative. Conversely, interactions achieving higher goals are generally less noisy, meaning $\varepsilon(a)$ is closer to a. To illustrate this, consider a scenario where a user clicks two recommended items (v_1 and v_2) under the same state. After clicking v_1, the user chooses to quit the system, while he stays longer and browses more items after clicking v_2. This indicates that the goal (i.e., cumulative reward) with v_2 is larger than that with v_1. In this case, we argue that v_2 should be considered as the user's truly preferred item over v_1. That's because the act of quitting, which results in a smaller goal, indicates user dissatisfaction with the previous recommendation, even though he did click v_1 before. Our proposed MOGCSL can model and leverage this mechanism based on multi-dimensional goals, which serve as a description of the **future effects of current actions on multiple objectives**. Specifically, by incorporating multi-dimensional goals as input, MOGCSL can effectively differentiate between noisy and noiseless samples in the training data. During inference, when high goals are specified as input, the model can make predictions based primarily on the patterns learned from the corresponding noiseless interaction data. It's worth noting that several prior studies on recommender system denoising [26, 27] have highlighted that such noises (i.e., users' noisy instant behaviors that don't align with their long-term, real interests) are very common and widely exist in real-world recommendation datasets.

To empirically validate this effect, we discuss relevant experiments in Appendix B.2 due to limited space. The results demonstrate the denoising capability of MOGCSL.

4 Experiments

In this section, we introduce our experiments to address the following research questions: 1) **RQ1.** How does MOGCSL perform when compared to previous methods for multi-objective learning in recommender systems? 2) **RQ2.** How does MOGCSL mitigate the complexity challenges encountered in prior research? 3) **RQ3.** How does the goal-generation strategies for inference perform?

4.1 Experimental Setup

Datasets We use two public datasets: Challenge15 and RetailRocket. Both of them include binary labels indicating whether a user clicked or purchased the currently recommended item from real world. More details of the datasets and implementation specifics are provided in Appendix B.

Baselines Prior research on multi-objective learning encompass both model structure adaptation and optimization constraints. We consider two representative model architectures: Shared-Bottom [4] and MMOE [4]. For optimization constraints, we compare four methods: Fixed-Weights [28] assigns fixed weights for different objectives based on grid search; DWA [5] dynamically adjusts weights by considering the dynamics of loss change; PE [11] generates Pareto-efficient recommendations across multiple objectives; Nash-MTL [29] characterize the Nash bargaining solution for multi-objective learning; FAMO [12] adjusts weights to achieve balanced task loss reduction in an efficient way.

Table 1: Comparison between MOGCSL and other baselines on RetailRocket and Challenge15.

[RetailRocket]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
Share-Fix	48.57±0.17	45.79±0.09	49.47±0.10	46.01±0.08	35.51±0.24	25.85±0.16	40.15±0.20	27.03±0.14
Share-DWA	48.11±0.04	45.83±0.08	48.64±0.08	45.96±0.06	34.20±0.27	25.53±0.12	38.43±0.31	26.60±0.13
Share-PE	48.67±0.12	45.94±0.03	49.42±0.04	46.13±0.02	35.69±0.08	**26.20±0.08**	40.28±0.11	**27.37±0.07**
Share-Nash	48.84±0.21	46.23±0.17	50.07±0.19	46.50±0.22	35.87±0.20	26.04±0.16	40.52±0.25	27.11±0.13
Share-FAMO	48.92±0.17	46.11±0.10	50.19±0.09	46.78±0.13	35.97±0.14	26.17±0.09	40.71±0.11	27.35±0.18
MMOE-Fix	47.74±0.09	44.01±0.05	48.61±0.11	44.23±0.04	35.29±0.16	25.67±0.09	40.04±0.26	26.87±0.11
MMOE-DWA	47.78±0.40	44.57±0.13	48.44±0.24	44.79±0.09	35.68±0.46	26.13±0.29	40.22±0.57	27.28±0.32
MMOE-PE	46.58±0.22	43.72±0.15	47.37±0.11	43.94±0.14	35.39±0.27	26.19±0.11	39.78±0.39	27.31±0.14
MMOE-Nash	47.88±0.27	46.49±0.29	51.17±0.22	47.10±0.31	35.79±0.18	26.02±0.27	41.03±0.20	26.39±0.28
MMOE-FAMO	47.93±0.32	46.42±0.23	51.24±0.19	47.15±0.21	35.92±0.21	26.14±0.17	41.15±0.39	26.62±0.14
RMTL	62.84±0.24	49.92±0.20	67.04±0.15	50.89±0.25	33.95±0.35	24.23±0.41	39.87±0.32	25.27±0.29
PMORS	63.14±0.15	51.02±0.17	67.45±0.20	52.07±0.13	34.16±0.26	24.09±0.23	39.86±0.19	25.31±0.27
MOPRL	61.18±0.19	50.74±0.10	64.76±0.25	51.65±0.02	33.99±0.11	24.31±0.08	38.98±0.08	25.57±0.08
MOGCSL	**65.43±0.15**	**52.92±0.11**	**69.28±0.14**	**53.90±0.14**	**36.30±0.25**	25.24±0.15	**41.92±0.35**	26.67±0.19

[Challenge15]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
Share-Fix	38.18±0.10	25.47±0.26	43.97±0.18	26.93±0.20	41.61±0.30	25.77±0.16	49.19±0.43	27.70±0.19
Share-DWA	38.27±0.18	25.63±0.08	43.95±0.19	27.07±0.08	41.49±0.24	25.90±0.16	48.90±0.20	27.77±0.14
Share-PE	38.92±0.09	25.83±0.13	44.82±0.12	27.32±0.07	42.46±0.16	26.39±0.06	50.05±0.17	28.32±0.06
Share-Nash	39.12±0.17	25.87±0.15	45.13±0.22	27.56±0.14	43.10±0.24	26.89±0.19	50.61±0.17	28.88±0.21
Share-FAMO	39.06±0.13	25.94±0.21	44.97±0.18	27.65±0.09	43.25±0.13	27.02±0.11	50.58±0.26	28.94±0.14
MMOE-Fix	35.34±0.12	23.87±0.07	40.68±0.09	25.22±0.12	43.82±0.16	27.33±0.09	51.42±0.19	29.26±0.10
MMOE-DWA	37.04±0.40	24.88±0.13	42.64±0.24	26.30±0.09	42.20±0.46	26.45±0.29	49.48±0.57	28.30±0.32
MMOE-PE	36.40±0.36	24.66±0.19	41.52±0.33	25.96±0.19	**44.04±0.09**	**27.44±0.03**	**51.60±0.07**	**29.37±0.03**
MMOE-Nash	38.01±0.41	25.40±0.32	43.69±0.27	27.06±0.29	43.77±0.41	26.87±0.52	50.96±0.59	28.92±0.50
MMOE-FAMO	37.92±0.56	25.43±0.22	43.63±0.29	27.12±0.15	43.71±0.55	26.98±0.38	50.98±0.66	29.03±0.39
RMTL	53.34±0.46	33.65±0.36	62.89±0.49	34.74±0.37	41.41±0.41	24.87±0.38	49.29±0.43	26.91±0.29
PMORS	54.98±0.31	34.77±0.28	63.52±0.24	37.05±0.35	42.36±0.39	25.10±0.42	50.07±0.30	26.84±0.26
MOPRL	54.79±0.37	35.37±0.26	63.10±0.45	37.49±0.27	42.14±0.21	25.62±0.18	50.18±0.25	27.66±0.19
MOGCSL	**56.82±0.25**	**35.93±0.15**	**65.64±0.55**	**38.17±0.19**	42.47±0.15	25.64±0.11	50.52±0.14	27.73±0.11

Following previous research[6], we consider all these optimization methods for each model architecture, resulting in eight baselines denoted as Share-Fix, MMOE-Fix, etc. Additionally, we introduce a variant of PRL [9], which firstly applied GCSL to recommender systems. Specifically, similar to classic multi-objective methods, we compute the weighted summation of rewards from different objectives at each timestep. Then the overall cumulative reward is calculated as the goal, which is a scalar following conventional GCSL. We call this variant as MOPRL. Since we formulate our problem as an MOMDP, we also incorporate a recent baseline RMTL [30], which applies offline reinforcement learning for sequential recommendation. Similarly, the aggregate reward is derived by the weighted summation of all objective rewards. Additionally, we compare against PMORS [31], a recent supervised multi-objective learning framework designed for recommender systems. Note that we employ the *T-enc* and self-attention block introduced in Section 3.2 as the **base module to encode sequential data for all compared baselines to ensure fair comparison**.

Evaluation Metrics We employ two widely recognized information retrieval metrics to evaluate model performance in top-k recommendation: Hit Ratio (HR@k) and Normalized discounted cumulative gain (NG@k).We report the mean and standard deviation over 5 seeds for each experiment.

4.2 Performance Comparison (RQ1)

We begin by comparing the performance of MOGCSL and selected baselines in terms of top-k recommendation, as reported in Table 1. It's worth mentioning that a straightforward strategy based on training set statistics is employed to determine the inference goals in PRL [9]. Specifically, at each timestep in inference, the goals are set as the average cumulative reward from offline data at the same timestep, multiplied by a hyper-parameter factor λ that is tuned using the validation set. To ensure a

fair comparison, we adopt the same strategy here for determining inference goals in MOGCSL. The comparison between different goal-choosing strategies is discussed in Section 4.4.

On RetailRocket, MOGCSL significantly outperforms previous multi-objective methods on purchase. Regarding click metrics, MOGCSL achieves the best performance on HR, while Share-PE slightly outperforms it on NDCG. However, the performance gap between Share-PE and MOGCSL for purchase-related metrics ranges from 17% to 20%, whereas Share-PE only marginally outperforms MOGCSL on NDCG for purchase by less than 1%. Additionally, we observe that the more complex architecture design, MMOE, can perform worse than the simpler Shared-Bottom in many cases. Surprisingly, a naive optimization strategy based on fixed loss weights can outperform more advanced methods like DWA across several metrics. These findings highlight the limitations of previous approaches that rely on assumptions about model architectures or optimization constraints, which may not necessarily be true in general environments. Similar trends are observed on Challenge15. While MMOE-PE performs slightly better on click by 1-2%, MOGCSL achieves a substantial performance improvement on the more important purchase metrics by 11-20%. Especially, PE optimizes for Pareto efficiency by assuming that improving one objective should not degrade performance on any other objective. However, this assumption does not align well with the realities of recommender systems. In practice, a significant improvement in high-value objectives (e.g., purchase) accompanied by only minor reductions in less critical objectives (e.g., click) is often preferable in practical scenarios. We've included more discussions on the relative performance in Appendix C.4 and C.5.

Apart from classic benchmarks for multi-objective learning, MOGCSL also exhibits significant and consistent performance improvements on both datasets compared to offline RL based RMTL and the variant MOPRL created on standard GCSL. Especially, at each timestep, the overall reward of them is calculated as the weighted sum of rewards across all objectives. In our experiments, it's defined as $r' = w_c r^c + w_p r^p$, where r^c and r^p are click and purchase reward and $w_c + w_p = 1$. Then the goal in MOPRL is derived by calculating the cumulative rewards as a scalar. In contrast, MOGCSL takes the goal as a vector, allowing the disentanglement of rewards for different objectives along different dimensions. Notably, no additional summation weights or other constraints are required. We have conducted experiments to compare the performance of MOGCSL to MOPRLs with different weight combinations. The results shown in Appendix B.5 demonstrate that MOGCSL consistently outperforms MOPRL across all weight combinations, demonstrating that representing the goal as a multi-dimensional vector enhances the effectiveness of GCSL on multi-objective learning.

4.3 Complexity Comparison (RQ2)

Apart from the performance improvement, MOGCSL also benefits from seamless integration with classic sequential models, adding minimal additional complexity. During the training stage, the only extra complexity arises from relabeling one-step rewards with goals and including them as input to the sequential model. In contrast, previous multi-objective learning methods often introduce significantly excess time and space complexity [3]. For instance, MMOE and Shared-Bottom both design separate towers for each objective [4], leading to a significant increase in model parameters as the number of tasks grows; RMTL needs an additional RL head for optimization on TD error. In contrast, MOGCSL only requires a simple MLP layer for action projection. In terms of time complexity, DWA, Nash-MTL and FAMO require recording and calculating loss change dynamics for each training epoch, while PE and PMORS involve computing the inverse of a large parameter matrix to solve an optimization problem under KKT conditions. Additionally, tuning the weight combinations for multiple objectives using grid search in methods like Fix-Weight and MOPRL is highly time-consuming, requiring approximately $O(m^{n-1})$ repetitive experiments to identify a near-optimal combination, where m is the size of the search space per dimension and n is the number of objectives. In contrast, MOGCSL inherently avoids this weight-tuning process. Table 2

Table 2: Time/Space complexity comparison.

	Model Size	Training time
Share-Fix	14.0M	9.6Ks
Share-DWA	14.0M	5.3Ks
Share-PE	14.0M	5.6Ks
Share-Nash	14.0M	8.6Ks
Share-FAMO	14.0M	5.1Ks
MMOE-Fix	14.1M	10.2Ks
MMOE-DWA	14.1M	9.5Ks
MMOE-PE	14.1M	60.5Ks
MMOE-Nash	14.1M	12.4Ks
MMOE-FAMO	14.1M	8.8Ks
RMTL	17.5M	100.2Ks
PMORS	14.2M	10.4Ks
MOPRL	9.1M	3.2Ks
MOGCSL	9.1M	3.0Ks

summarizes the comparison on RetailRocket, where MOGCSL significantly benefits from a smaller model size and faster training speed. See more discussions of complexity in Appendix C.1.

4.4 Goal-generation Strategy Comparison (RQ3)

As introduced in Section 3.3, most previous research on GCSL decides the inference goals based on simple statistics on the training set. However, we demonstrate that the distribution of the goals achieved by the agent during inference should be jointly determined by the initial state, input goal and behavior policy. Based on that, we propose a novel algorithm (see Algorithm 2) that leverages CVAE to derive feasible and desirable goals for inference. Note that an utility principle $U(\boldsymbol{g})$ is required to evaluate the goodness of the multi-dimensional goals, which is generally preferable for "high" goals but could be flexible with specific business requirements. In our experiments, we select the best goal $\tilde{\boldsymbol{g}}_b^a$ from the set $\boldsymbol{G}^a$ based on the following rule, which ensures that no goal within the achievable set is superior to the selected goal across all objectives:

$$\tilde{\boldsymbol{g}}_b^a = \tilde{\boldsymbol{g}} \in \boldsymbol{G}^a, \text{ s.t. } \nexists \tilde{\boldsymbol{g}}' \in \boldsymbol{G}^a \setminus \tilde{\boldsymbol{g}}, \ \tilde{g}'_i \geq \tilde{g}_i \ \forall i \in [1, d]. \tag{8}$$

We compare two variants of MOGCSL here. MOGCSL-S employs the statistical strategy introduced in Section 4.2, while MOGCSL-C utilizes the goal-choosing method based on CVAE. As shown in Table 3, we observe that these two strategies do not significantly differ in overall performance across both datasets. While MOGCSL-C performs better on RetailRocket, it exhibits worse performance on Challenge15. To investigate the reason, we conduct an additional experiment by varying the factor λ for the inference goals of MOGCSL-S. The results reveals that the optimal performance is achieved when the factor is set between 1 and 2 for all metrics. When it grows larger, performance consistently declines. The figure is shown in Appendix B.6. Similar findings have been reported in prior research [8, 9, 22], demonstrating that setting very large inference goals can indeed harm performance.

We posit that the sparsity of training data within the high-goal space may contribute to the suboptimal performance of more advanced goal-choosing methods. While we may find some potentially achievable high goals, the model lacks sufficient training data to learn effective actions to reach these goals. Notably, the mean cumulative reward across all trajectories in both datasets is only around 5.3 for click and 0.2 for purchase. Consequently, most training data demonstrates how to achieve relatively low goals, hindering the model's ability to generalize effectively for larger goals in inference. To validate this, we conduct additional experiments on another dataset, Tenrec, with higher average goals. The two goals are defined for click and like behaviors. The mean cumulative rewards across all trajectories are approximately 28.3 for click and 1.2 for like, both notably higher than those observed in the RetailRocket and Challenge15 datasets. The results in Table 3 show that MOGCSL-C significantly outperforms MOGCSL-S on Tenrec. Together with previous experiments, it shows that insufficient high-quality trajectories in offline datasets can lead to suboptimal outcomes when GCSL policies are overly optimistic.

It's worth noting that a very recent work, RVDT [32], provides an initial attempt at offering a more formal and theoretically grounded analysis of this problem. Leveraging insights and relevant theory from offline reinforcement learning, the authors draw similar conclusions and demonstrate that insufficient high-quality trajectories in offline datasets can lead to suboptimal outcomes when GCSL policies are overly optimistic.

Applying these insights to our setting, although the CVAE-based algorithm can identify potentially achievable high-value goals, the GCSL model may lack sufficient training data to learn effective actions for reaching these goals. Consequently, a more balanced strategy, such as selecting goals based on the average cumulative goals in the offline data, scaled by a hyperparameter factor, often proves more effective when high-goal training data is limited.

Importantly, we'd like to highlight that this doesn't weaken the framework's practical value. MOGCSL-S already outperforms existing baselines with low inference cost, while MOGCSL-C provides further gains when richer high-goal data is available. We therefore view the two variants as complementary: MOGCSL-S is a strong default choice, whereas MOGCSL-C is preferable in less sparse, higher-goal regimes. Thus, rather than limiting the practicality, this observation helps for appropriate strategy selection tailored to practical use cases. See more discussion in Appendix C.7.

The results provide several valuable insights for selecting goal-choosing strategies for MOGCSL. First, strategies based on simple statistics on the training data prove to be efficient and effective in

Table 3: Comparison between statistical strategy and CVAE-based method for goal-choosing.

[RetailRocket]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MOGCSL-S	**65.43**$\pm$**0.15**	**52.92**$\pm$**0.11**	69.28$\pm$0.14	53.90$\pm$0.14	36.30$\pm$0.25	25.24$\pm$0.15	41.92$\pm$0.35	26.67$\pm$0.19
MOGCSL-C	65.01$\pm$0.07	52.89$\pm$0.04	**69.34**$\pm$**0.05**	**54.00**$\pm$**0.04**	**36.54**$\pm$**0.02**	**25.41**$\pm$**0.04**	**42.20**$\pm$**0.03**	**26.84**$\pm$**0.06**

[Challenge15]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MOGCSL-S	**56.82**$\pm$**0.25**	**35.93**$\pm$**0.15**	**65.64**$\pm$**0.55**	**38.17**$\pm$**0.19**	**42.47**$\pm$**0.15**	**25.64**$\pm$**0.11**	**50.52**$\pm$**0.14**	**27.73**$\pm$**0.11**
MOGCSL-C	55.13$\pm$0.07	35.04$\pm$0.02	63.98$\pm$0.04	37.30$\pm$0.03	42.14$\pm$0.04	25.37$\pm$0.07	50.12$\pm$0.09	27.53$\pm$0.05

[Tenrec]	Like (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MOGCSL-S	5.96$\pm$0.17	2.15$\pm$0.12	6.93$\pm$0.20	2.70$\pm$0.11	4.87$\pm$0.13	1.52$\pm$0.08	5.67$\pm$0.14	1.95$\pm$0.11
MOGCSL-C	**6.78**$\pm$**0.07**	**2.99**$\pm$**0.02**	**7.84**$\pm$**0.05**	**3.86**$\pm$**0.04**	**5.66**$\pm$**0.05**	**2.14**$\pm$**0.05**	**6.73**$\pm$**0.03**	**2.68**$\pm$**0.07**

most cases, particularly when low computation complexity is required during inference. Second, if we aim to further enhance performance using more advanced goal-choosing algorithms, access to a training set with more instances with high-valued goals could be crucial. Last, our MOGCSL proposes to address and model the achievability of multi-objective goals at inference. Within this framework, we explore and validate both principled (MOGCSL-C) and heuristic (MOGCSL-S) goal-selection methodologies to derive these achievable and "high" goals on multiple objectives. These methods demonstrate complementary strengths across diverse datasets and empirical conditions, which is itself a non-trivial finding that is valuable for practical application of MOGCSL.

5 Conclusion

In this work, we propose a novel framework named MOGCSL for multi-objective recommendation. MOGCSL utilizes a vectorized goal to disentangle the representation of different objectives. It can be directly combined with conventional sequential models and optimized through supervised learning, without requiring handcrafted model architecture or optimization constraints. Beyond training process, we theoretically analyze the properties of inference goals and propose a novel goal-generation algorithm accordingly. Extensive experiments demonstrate the superiority of MOGCSL in both effectiveness and efficiency. Future work can extend MOGCSL to other application scenarios.

Despite its effectiveness and efficiency, MOGCSL has several limitations that motivate future work. These include improving CVAE-based goal selection when high-goal trajectories are sparse through data augmentation or iterative online training, explicitly incorporating negative feedback into goal modeling, and validating practical benefits through live-user A/B testing. Extending the framework beyond recommendation to other multi-objective learning tasks is another promising direction. We've include more detailed discussions of limitation and future works in Appendix D.

6 Acknowledgment

This work was supported by the Intuit University Collaboration Program.

References

[1] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[2] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[3] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

[4] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939, 2018.

[5] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019.

[6] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[7] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and Zhaochun Ren. Rethinking reinforcement learning for recommendation: A prompt perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1347–1357, 2022.

[10] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[11] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, and Peng Jiang. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender systems*, pages 20–28, 2019.

[12] Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[14] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[15] Zifeng Zhuang, Dengyun Peng, Jinxin Liu, Ziqi Zhang, and Donglin Wang. Reinformer: Max-return sequence modeling for offline rl. *arXiv preprint arXiv:2405.08740*, 2024.

[16] Qingpeng Cai, Ruohan Zhan, Chi Zhang, Jie Zheng, Guangwei Ding, Pinghua Gong, Dong Zheng, and Peng Jiang. Constrained reinforcement learning for short video recommendation. *arXiv preprint arXiv:2205.13248*, 2022.

[17] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 957–965, 2022.

[18] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48: 67–113, 2013.

[19] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478*, 2022.

[20] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[21] Chengxi Li, Yejing Wang, Qidong Liu, Xiangyu Zhao, Wanyu Wang, Yiqi Wang, Lixin Zou, Wenqi Fan, and Qing Li. Strec: Sparse transformer for sequential recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 101–111, 2023.

[22] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[24] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[25] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[26] Kaike Zhang, Qi Cao, Fei Sun, Yunfan Wu, Shuchang Tao, Huawei Shen, and Xueqi Cheng. Robust recommender system: a survey and future directions. *ACM Computing Surveys*, 58(1): 1–38, 2025.

[27] Jiawei Chen, Xiang Wang, Fuli Feng, and Xiangnan He. Bias issues and solutions in recommender system: Tutorial on the recsys 2021. In *Proceedings of the 15th ACM conference on recommender systems*, pages 825–827, 2021.

[28] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.

[29] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

[30] Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, et al. Multi-task recommendations with reinforcement learning. In *Proceedings of the ACM web conference 2023*, pages 1273–1282, 2023.

[31] Jipeng Jin, Zhaoxiang Zhang, Zhiheng Li, Xiaofeng Gao, Xiongwen Yang, Lei Xiao, and Jie Jiang. Pareto-based multi-objective recommender system with forgetting curve. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4603–4611, 2024.

[32] Wensong Bai, Chufan Chen, Yichao Fu, Qihang Xu, Chao Zhang, and Hui Qian. Rebalancing return coverage for conditional sequence modeling in offline reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[33] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[34] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 931–940, 2020.

[35] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[36] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 357–358, 2015.

[37] Roman Zykov, Noskov Artem, and Anokhin Alexander. Retailrocket recommender system dataset, 2022. URL `https://www.kaggle.com/dsv/4471234`.

[38] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. *Advances in Neural Information Processing Systems*, 35:11480–11493, 2022.

A Proof of Theorem 1

Proof. Let $H := |\tau|$ be the length of the trajectory, and write

$$\tau = (\boldsymbol{s}_1, a_1, r_1, \ldots, \boldsymbol{s}_H, a_H, r_H, \boldsymbol{s}_{H+1}).$$

Fix the MOMDP environment, the initial state s_1, the initial goal g_1, and the goal-conditioned policy $\pi(a \mid \boldsymbol{s}, \boldsymbol{g})$.

For each timestep t, the goal input to the policy is updated according to

$$\boldsymbol{g}_{t+1} = \boldsymbol{g}_t - r_t,$$

or equivalently,

$$\boldsymbol{g}_t = \boldsymbol{g}_1 - \sum_{i=1}^{t-1} r_i, \qquad t = 1, \ldots, H,$$

where the empty sum is defined as 0. Hence, $\boldsymbol{g}_t$ is a deterministic function of $(\boldsymbol{g}_1, r_1, \ldots, r_{t-1})$.

By the chain rule, the conditional distribution of the full trajectory is

$$\Pr(\tau \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi) = \prod_{t=1}^{H} \Pr(a_t, r_t, \boldsymbol{s}_{t+1} \mid \boldsymbol{s}_{1:t}, a_{1:t-1}, r_{1:t-1}, \boldsymbol{g}_1, \pi).$$

Since the policy chooses actions according to the current state and current goal, and since the environment is an MOMDP with transition probability $T(\boldsymbol{s}' \mid \boldsymbol{s}, a)$ and reward distribution $P(r \mid \boldsymbol{s}, a)$, we have

$$\Pr(a_t, r_t, \boldsymbol{s}_{t+1} \mid \boldsymbol{s}_{1:t}, a_{1:t-1}, r_{1:t-1}, \boldsymbol{g}_1, \pi) = \pi(a_t \mid \boldsymbol{s}_t, \boldsymbol{g}_t)\, P(r_t \mid \boldsymbol{s}_t, a_t)\, T(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t).$$

Therefore,

$$\Pr(\tau \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi) = \prod_{t=1}^{H} \pi(a_t \mid \boldsymbol{s}_t, \boldsymbol{g}_t)\, P(r_t \mid \boldsymbol{s}_t, a_t)\, T(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t).$$

Substituting

$$\boldsymbol{g}_t = \boldsymbol{g}_1 - \sum_{i=1}^{t-1} r_i$$

gives

$$\Pr(\tau \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi) = \prod_{t=1}^{H} \pi\left(a_t \,\middle|\, \boldsymbol{s}_t,\, \boldsymbol{g}_1 - \sum_{i=1}^{t-1} r_i\right) P(r_t \mid \boldsymbol{s}_t, a_t)\, T(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t).$$

Because the environment is fixed, both $T(\cdot \mid \boldsymbol{s}, a)$ and $P(\cdot \mid \boldsymbol{s}, a)$ are fixed. Hence, the distribution of the full trajectory τ is completely determined by $(\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$.

Now define

$$\boldsymbol{g}^a(\tau) := \sum_{t=1}^{H} r_t.$$

Since $\boldsymbol{g}^a(\tau)$ is a deterministic function of the trajectory τ, for any measurable set $B \subseteq \mathbb{R}^d$, we have

$$\Pr(\boldsymbol{g}^a \in B \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi) = \int \mathbf{1}\left\{\sum_{t=1}^{H} r_t \in B\right\} \Pr(d\tau \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi).$$

Therefore, the distribution of $\boldsymbol{g}^a$ is completely determined by the distribution of τ. Since we have shown that $\Pr(\tau \mid \boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$ is determined by $(\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$, it follows that the distribution of $\boldsymbol{g}^a$ is also determined by $(\boldsymbol{s}_1, \boldsymbol{g}_1, \pi)$.

This proves Theorem 1. □

Remark. If the session length is random, the same argument still applies by introducing an absorbing terminal state with zero reward after termination, or equivalently by integrating over all possible finite trajectory lengths.

B Experiment Details

B.1 Model Structure

The model structure of MOGCSL is shown in Figure 1.

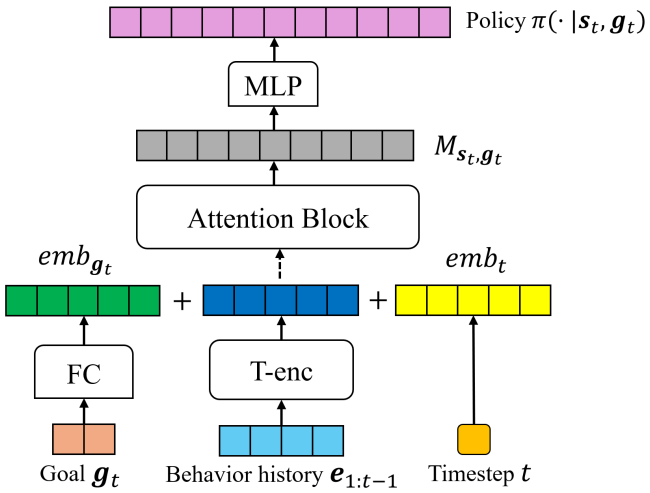

Figure 1: Model structure of MOGCSL.

B.2 Denoising Capability Experiments

To illustrate the denoising capability of MOGCSL, we consider the same set-up introduced in Section 3.4, including definitions and notation of state, action, and reward. As described before, we assume that the noisy portion of the training data originates from users who are presented with a list of items that are not suitable for them, rendering their choices for these recommendations not meaningful. Conversely, data samples with higher goals are generally less noisy, meaning $\varepsilon(a)$ is closer to a. We also record a long-term and multidimensional goal (e.g., the cumulative reward) $\boldsymbol{g} = (g_1, ..., g_n)$ at each interaction (known only at the end of the session, but recorded retroactively). Thus our training data is a sample from a distribution $\mathcal{D}$ of tuples $(\boldsymbol{s}, \boldsymbol{g}, \varepsilon(a))$, where the state of the user is represented by a vector $\boldsymbol{s}$.

As far as we know, in current real-world recommendation datasets, there are **no instance-level labels that distinguish random or weakly preference-driven interactions from genuinely preference-**

Table 4: Performance and relative performance drop by training on corrupted, noisy RetailRocket dataset.

	Purchase (%)				Click (%)			
	HR@20	Drop Ratio	NG@20	Drop Ratio	HR@20	Drop Ratio	NG@20	Drop Ratio
Share-FAMO	46.32±0.11	7.71↓	42.33±0.27	9.51↓	36.97±0.24	9.19↓	25.14±0.17	8.08↓
MMOE-FAMO	47.01±0.19	8.26↓	41.92±0.18	11.09↓	37.41±0.17	9.09↓	24.83±0.23	6.72↓
RMTL	63.21±0.09	5.71↓	47.60±0.27	6.46↓	36.92±0.17	7.40↓	23.82±0.15	5.74↓
PMORS	64.17±0.18	4.86↓	49.01±0.16	5.88↓	36.05±0.30	9.56↓	23.89±0.20	5.61↓
MOPRL	62.37±0.11	3.69↓	48.98±0.14	5.17↓	36.61±0.21	6.08↓	24.30±0.17	4.97↓
MOGCSL	**67.92**±0.17	**1.96**↓	**52.13**±0.21	**3.28**↓	**39.91**±0.16	**4.79**↓	**25.85**±0.20	**3.07**↓

Table 5: Comparison of XGBoost with different inputs. The mean and standard deviation over 5 seeds are reported.

	Accuracy	M-Logloss
XGBoost-s	0.0576±0.0038	3.7595±0.0393
XGBoost-ug	0.0598±0.0009	3.7262±0.0281
XGBoost-mg	**0.0634±0.0027**	**3.6603±0.0097**

reflective ones. For this reason, we use controlled noise injection and synthetic experiments to explicitly study this mechanism.

We first evaluate robustness on the real-world dataset, RetailRocket, by explicitly injecting noise into the training data. Specifically, we first randomly sample 20% of the original training sequences from those with the lowest 20% average goal (i.e., cumulative reward) computed over the click and purchase objectives. For each selected sequence, we replace its final ground-truth item with a uniformly random item from the catalog. This procedure introduces mislabeled interactions that are associated with low long-term goals, which fail to reflect users' true preferences and correspond to our noise definition.

We then train the baseline methods, Share-FAMO, MMoE-FAMO, RMTL, PMORS, MOPRL, and our proposed MOGCSL, on this corrupted training set, while keeping evaluation unchanged on the original test set.

Table 4 reports performance under noisy training as well as the relative performance drop compared to training on the original data. Across metrics, MOGCSL exhibits the smallest degradation, indicating substantially higher robustness to low-goal interaction noise. In contrast, the baselines suffer larger performance drops, suggesting greater sensitivity to noisy user behaviors that are weakly aligned with long-term objectives.

Besides the noise injection experiments, we further conduct experiments to study MOGCSL's denoising capability in a more fine-grained way. As far as we know, there is no real-world recommendation dataset that contains the labels distinguishing between random and genuinely preference-driven user interactions. As a result, we generate a dataset to study the effect as follows: (1): The states and the goals are sampled from two independent multivariate normal distributions with dimension of 50 and 5 respectively (mean is 0 and variance is 2 in each dimension). (2) The ground-truth action a is entirely determined by $\boldsymbol{s}$, whose ID is set to the number of coordinates of $\boldsymbol{s}$ that are greater than 0. (3) Define $\varepsilon(a)$ as: $\varepsilon(a) = a$ if $g_i > -1$ for all i; otherwise $\varepsilon(a)$ is uniformly random.

Formally, $\varepsilon(a)$ is determined by $\boldsymbol{g}$ and $\boldsymbol{s}$ as follows:

$$\varepsilon(a) = (\prod_{i=0}^{n-1} \mathbb{1}(g_i > -1))a + (1 - \prod_{i=0}^{n-1} \mathbb{1}(g_i > -1))randint[0, N-1], \tag{9}$$

where $a = \sum_{j=0}^{N-1} \mathbb{1}(s_j > 0)$, N is the dimension of states and n is the dimension of goals.

Since MOGCSL is applicable to any supervised model by integrating goals as additional input features, we choose a simple XGBoost classifier [33] here for the sake of clarity. Specifically, we train three variants of XGBoost classifier on this data to predict the action given a state: (1) XGBoost-s: this variant only takes $\boldsymbol{s}$ as input and ignores $\boldsymbol{g}$. It cannot detect noisy instances in $\mathcal{D}$ because it lacks access to $\boldsymbol{g}$. (2) XGBoost-ug: this variant is taken as a single-objective GCSL model, which takes $\boldsymbol{s}$ and only the first coordinate g_1 of $\boldsymbol{g}$ as input. Clearly, it's also unable to precisely distinguish noisy data since $\varepsilon(a)$ is determined by all dimensions of $\boldsymbol{g}$ (as shown in Eq. (9)). (3) XGBoost-mg: this variant is based on our multi-objective GCSL, which takes both $\boldsymbol{s}$ and $\boldsymbol{g}$ as input. It is the only one capable of distinguishing all the noisy data by learning the determination pattern from $\boldsymbol{g}$ and $\boldsymbol{s}$ to $\varepsilon(a)$.

During inference stage, for XGBoost-ug and XGBoost-mg, we adopt a simple strategy to determine the goals: directly setting each dimension of $\boldsymbol{g}$ to 1, which serves as a high value to satisfy the condition $g_i > -1$ for the noiseless samples where $\varepsilon(a) = a$.

The results are presented in Table 5. It is evident that XGBoost-mg achieves the best performance. By incorporating multi-dimensional goals as input, XGBoost-mg can effectively differentiate between noisy and noiseless samples in the training data based on MOGCSL. During inference, when a high

goal is specified as input, the model can make predictions based solely on the patterns and knowledge learned from the noiseless data.

In summary, MOGCSL demonstrates a validated inherent denoising capability, as evidenced by its superior empirical performance on both synthetic and real-world industrial datasets.

Furthermore, we'd like to highlight that the injected-noise and synthetic experiments are **controlled validations** of the specific denoising mechanism studied here. By contrast, the main experiments discussed in Section 4.2 are **conducted on real-world datasets without manually injected noise**. In those experiments, MOGCSL consistently outperforms strong MTL baselines, including recommendation-specific ones like RMTL and PMORS, which don't explicitly model this type of noise. This is the evidence that the proposed mechanism is useful in realistic recommendation settings.

Our denoising claim is scoped to noise associated with low-goal trajectories, rather than arbitrary goal-independent noise. As discussed in Section 3.4, some short-term interactions in recommendation systems are weak indicators of true user preference when they are not followed by meaningful long-term engagement. Prior work [26, 27] similarly notes that such noisy short-term behaviors are common in real logged recommendation data. Our controlled experiments were therefore deliberately designed so that lower-goal trajectories contain more noise.

We therefore emphasize that the appendix experiments are not intended to establish robustness to arbitrary goal-independent noise. That broader setting is different from the denoising scenario we address here. Our claim is not that MOGCSL is a universal denoising method, but that it is particularly effective when noisier supervision is correlated with lower long-term outcomes.

B.2.1 Datasets

We conduct experiments on two publicly available datasets: Challenge15 [1] and RetailRocket [2]. They are both collected from online e-business platforms by recording users' sequential behaviours in recommendation sessions. Specifically, both datasets include binary labels indicating whether a user clicked or purchased the currently recommended item. Following previous research [9, 34], we filter out sessions with lengths shorter than 3 or longer than 50 to ensure data quality.

After preprocessing, the Challenge15 dataset comprises 200,000 sessions, encompassing 26,702 unique items, 1,110,965 clicks and 43,946 purchases. Similarly, the processed RetailRocket dataset consists of 195,523 sessions, involving 70,852 distinct items. It documents 1,176,680 clicks and 57,269 purchases. We partition them into training, validation, and test sets, maintaining an 8:1:1 ratio.

B.2.2 Baseline Details

In the experiments, we compare two representative model architectures for multi-objective learning:

- **Shared-Bottom** [4]: A classic model structure for multi-objective learning. The bottom of the model is a neural network shared across all objectives. On top of this shared base, separate towers are added for each objective, producing predictions specific to that objective.
- **MMOE** [4]: A widely used multi-objective model architecture. It first maps inputs to multiple expert modules shared by all objectives. These experts contribute to each objective through designed gates. The final input for each tower is a weighted summation of the experts' outputs.

Beyond architectural adaptations, other works focus on studying optimization constraints, mainly through adjusting weights of losses for different objectives. We compare the following methods:

- **Fixed-Weights** [28]: A straightforward strategy that assigns fixed weights based on grid search results from the validation set. These weights remain constant throughout the whole training stage.

[1] https://recsys.acm.org/recsys15/challenge

[2] https://www.kaggle.com/retailrocket/ecommerce-dataset

- **DWA** [5]: It's designed to dynamically assign weights by considering the rate of loss change for each objective during recent training epochs. It tends to assign larger weights to objectives with slower loss changes.

- **PE** [11]: It's designed for generating Pareto-efficient recommendations across multiple objectives. It's optimized for Pareto efficiency, ensuring no further improvement in one objective comes at the expense of any others.

- **Nash-MTL** [29]: This method frames the gradient combination step in multi-objective learning as a bargaining game and use the Nash bargaining solution to find the optimal update direction.

- **FAMO** [12]: This recent method aims to dynamically adjust the weights for multi-objective learning, achieving balanced task loss reduction while maintaining relatively low space and time complexity.

Note that to ensure a fair comparison, we employ the *T-enc* and self-attention block introduced in Section 3.2 as the base module to encode sequential data for all compared baselines. This ensures that user sequential behavior information is captured consistently in a concise and computationally efficient manner, aligning with our core motivation of designing a lightweight, scalable, and efficient framework for multi-objective recommendation.

B.3 Evaluation Metrics

We employ two widely recognized information retrieval metrics to evaluate model performance in top-k recommendation. Hit Ratio (HR@k) is to quantify the proportion of recommendations where the ground-truth item appears in the top-k positions of the recommendation list [35]. Normalized discounted cumulative gain (NDCG@k) further considers the positional relevance of ranked items, assigning greater importance to top positions during calculation [20]. Given our dual objectives in experiments, we evaluate performance using HR@k and NDCG@k based on corresponding labels for click and purchase events (i.e., whether an item was clicked or purchased by the user).

B.4 Implementation Details

First, to ensure a fair comparison, we employ the transformer encoder and self-attention block introduced in Section 3.2 as the base module to encode the input features for all compared baselines. We preserve the 10 most recent historical interaction records to construct the state representation. For sequences shorter than 10 interactions, we pad them with a padding token. The embedding dimensions for both state and goal are set to 64, and the batch size is fixed at 256. We utilize the Adam optimizer for all models, tuning the learning rate within the range of [0.0001, 0.0005, 0.001, 0.005]. Additionally, for methods that necessitate manual assignment of weights, we fine-tune these weights in the range of [0.1, 0.2, ..., 0.9] based on their performance on the validation set. The sample size K in Algorithm 2 is set to 20 in the experiments. For all baseline methods, we adhere to the implementation details, optimization configurations, and hyperparameter tuning strategies reported in their original papers. All experiments are conducted five times, each with different random seeds, and we report the mean and standard deviation of the results. When comparing the time complexity, each experiment was conducted with a separate NVIDIA RTX 3090 and AMD 3960X.

B.5 Comparison between MOGCSL and MOPRLs

Figure 2 shows the performance comparison of MOGCSL to MOPRLs with different weight combinations.

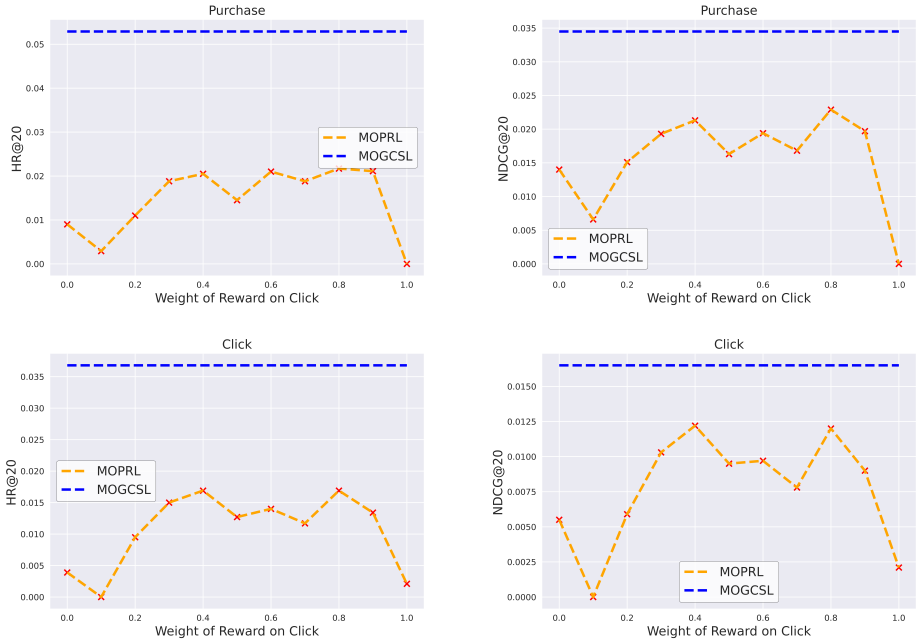

Figure 2: Comparison between MOGCSL and MOPRLs with different weight combinations on RetailRocket. Performance of MOGCSL is not dependent on the weights.

B.6 Comparison of MOGCSL-S w.r.t Factors

The detailed results of the performance comparison of MOGCSL-S with different factors for inference goal is shown in Figure 3.



Figure 3: Performance of MOGCSL-S with different factors for inference goal.

C Additional Discussions

C.1 Discussion on Complexity

We have provided a detailed comparison of the space complexity and training cost of MOGCSL relative to other baselines, demonstrating its advantages in terms of smaller model size and faster training speed. Here, we give a further examination of the computational complexity of MOGCSL at inference time, as well as the effect of the CVAE module.

As outlined in Section 3.2, MOGCSL employs a simple policy network architecture built from basic attention modules combined with simple fully connected layers. Consequently, inference through the policy network incurs minimal computational cost, as further evidenced by its smaller model size compared to architectures such as MMOE or reinforcement learning frameworks (see Table 2). For the CVAE-based goal-selection module, we also adopt a lightweight MLP architecture with approximately 1.6M parameters, which converges rapidly during training (within 6 epochs). We've compared the training time of MOGCSL with the CVAE module (MOGCSL-C) with other baselines. As demonstrated in Table 6, we can see that even with the additional computation introduced by the CVAE, MOGCSL-C remains among the most efficient multi-objective learning methods.

At inference, forward propagation through this CVAE introduces only negligible overhead due to the simplicity of the MLP structure. Although goal sampling introduces additional computation, it is required only once at the beginning of each inference sequence in practical applications. Subsequent goals are computed efficiently through iterative subtraction of current rewards, as established in our theoretical analysis.

Another important advantage of MOGCSL lies in its scalability to higher-dimensional objectives. By representing multiple objectives as a single vector input, MOGCSL naturally integrates additional goals without increasing architectural complexity. This vectorized formulation enhances scalability relative to conventional multi-objective approaches, which typically experience rapidly growing complexity with dimensionality. Importantly, this design eliminates the need for architectural modifications or substantial computational overhead when scaling to higher dimensions, as demonstrated in Table 2.

Table 6: Comparison of time and space complexity of MOGCSL-C on RetailRocket.

	Model Size	Training time
Share-Fix	14.0M	9.6Ks
Share-DWA	14.0M	5.3Ks
Share-PE	14.0M	5.6Ks
Share-Nash	14.0M	8.6Ks
Share-FAMO	14.0M	5.1Ks
MMOE-Fix	14.1M	10.2Ks
MMOE-DWA	14.1M	9.5Ks
MMOE-PE	14.1M	60.5Ks
MMOE-Nash	14.1M	12.4Ks
MMOE-FAMO	14.1M	8.8Ks
RMTL	17.5M	100.2Ks
PMORS	14.2M	10.4Ks
MOPRL	9.1M	3.2Ks
MOGCSL-C	10.7M	4.8Ks

C.2 Discussion on Learning Paradigm

MOGCSL is designed for automated and effective learning in multi-objective settings by leveraging offline trajectories conditioned on successfully achieved multi-dimensional goals. The key intuition is to reframe the challenge of balancing potentially conflicting objectives into the more tractable task of learning to achieve realistic goals directly from observed trajectories. During training, MOGCSL directly learns to reach certain goals across multiple objectives by utilizing each trajectory in the training data as a demonstration of successful achievement of the goal that it actually achieves.

At inference, MOGCSL models the distribution of achievable goals across multiple objectives conditioned on the initial state. From this distribution, it samples goals that are both realistic and aligned with practical business requirements, as specified through a customizable utility function. By always grounding goal selection in attainable outcomes, MOGCSL naturally respects inter-objective trade-offs and avoids the need for manually imposed constraints or heuristic weight tuning. This design yields a more robust and flexible framework for multi-objective optimization, readily adaptable to diverse application scenarios.

Specifically, the utility principle used in our experiments is defined in Eq. (8), which selects goals in a moderate manner such that no other achievable goal dominates it across all objectives. This design aligns with our aim of producing balanced and good performance across multiple objectives

in the experiments. However, this utility principle can also be flexibly adapted to prioritize different objectives based on application needs in MOGCSL. For instance, if one objective holds greater business importance, practitioners can modify the partial ordering relationship accordingly. One of the simplest and most direct ways could be assigning higher weights to emphasize it. However, we note that this weighting mechanism differs fundamentally from loss weighting in training-based methods (e.g., MOPRL and DWA in our baselines). In MOGCSL, training is not guided by predefined objective weights; instead, the model learns policies to achieve diverse goals across all objectives demonstrated by offline trajectories. The relative importance of objectives is only applied during inference for goal selection among achievable goals, allowing the policy to flexibly align with empirical requirements. This inherent flexibility enables practitioners to tailor MOGCSL to diverse real-world scenarios and evolving objective priorities with minimal retraining effort.

C.3 Discussion on Contribution and Novelty

To further clarify the contribution and impact of our research, we summarize several important innovations that distinctly address critical challenges in general goal-conditional supervised learning and multi-objective recommendation scenarios:

- We are the first to integrate an MOMDP approach into GCSL's framework, extending the goal from a scalar to a multidimensional vector. This extension is non-trivial since defining and selecting achievable "high" multi-dimensional goals is conceptually complex and previously unexplored. Our paper proposes and evaluates both principled and heuristic goal-selection approaches, demonstrating their effectiveness across different datasets.
- We provide the first theoretical characterization of achievable goal distributions within the multi-objective setting of GCSL, formalized through Theorem 1 with complete proof (Section 3.3 and Appendix A).
- Building upon this theoretical foundation, we developed a novel goal-choosing algorithm that models the distribution of achievable goals over interaction sequences to select desirable high goals for inference. This algorithm is novel even for the single-objective case and addresses a fundamental challenge in GCSL applications.
- We were the first to demonstrate GCSL's capability to mitigate the harmful effects of noisy instances, which are common in real-world recommendation data. We showed that MOGCSL can leverage the future impact of current actions across multiple objectives to achieve this ability (Section 3.4 and Appendix B.2).

The novelty of this work is not simply replacing a scalar goal with a vector. Rather, our contribution is to **extend GCSL to the multi-objective setting and address the inference-time challenge that then arises**: how to define and select achievable high goals across multiple objectives.

This issue is largely absent in scalar-goal GCSL. In the scalar setting, inference-time goal selection can often be handled with simple heuristics or validation-based tuning, since only a single target value needs to be searched; this is exactly what standard GCSL approaches do, as discussed in Section 3.3. In the multi-objective setting, however, the goal becomes a vector of coupled objectives, and naive tuning can become impractical as dimensionality grows. Moreover, collapsing multiple objectives into a scalar requires manually chosen weights, which impose a fixed trade-off and discard the original per-objective structure.

Our use of vector-valued goals is therefore not merely representational. It preserves the objectives and makes achievability modeling and principled goal selection essential. Our framework addresses this by modeling achievable-goal distributions and proposing inference-time goal-selection strategies. To the best of our knowledge, this is **the first work to study this issue systematically in GCSL for multi-objective learning**. Extending scalar goals to vector-valued goals may appear straightforward, but modeling achievability and selecting goals jointly across multiple dimensions are far from trivial.

C.4 Discussion on Relative Performance

As shown and discussed in Section 4.4, MMOE-PE optimizes for strict Pareto efficiency by assuming that improving one objective should not degrade performance on any other objective. Under this criterion, MMOE-PE achieves slightly better click performance than MOGCSL on Challenge15,

Table 7: Comparison between PE and MOGCSL on Tenrec dataset.

	Like (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MMOE-PE	5.38±0.22	1.94±0.19	6.14±0.35	2.47±0.28	4.81±0.18	**1.59±0.20**	5.60±0.19	**1.98±0.23**
MOGCSL-S	**5.96±0.17**	**2.15±0.12**	**6.93±0.20**	**2.70±0.11**	**4.87±0.13**	1.52±0.08	**5.67±0.14**	1.95±0.11

with gains of about 1-2%. However, MOGCSL delivers much larger improvements on the more important purchase metrics, with gains of 11-20%. This suggests that, on the Challenge15 dataset, optimizing under a strict Pareto-efficiency constraint severely limits performance on the valuable purchase objective.

We then analyze this phenomenon from the perspective of dataset characteristics. In particular, we compute the Pearson correlation between click and purchase signals on both RetailRocket and Challenge15. The correlation is 0.72 on RetailRocket, but only 0.33 on Challenge15. This indicates that click and purchase behaviors are much more aligned on RetailRocket, whereas they exhibit substantially greater conflict on Challenge15. As a result, while MMOE-PE can maintain a slight advantage on click-related metrics, its strict Pareto-efficiency requirement restricts its ability to optimize purchase performance on Challenge15. By contrast, MOGCSL does not rely on a strict Pareto-efficiency constraint in either learning or objective decision, which enables a more practically useful trade-off across objectives. In recommendation settings, a substantial improvement on high-value objectives (e.g., purchase) with only minor reductions on less critical objectives (e.g., click), is often preferable in real-world applications.

To further investigate this phenomenon, we also compare MOGCSL with MMOE-PE on Tenrec, where the Pearson correlation between like and click is 0.52, indicating an intermediate level of objective alignment between Challenge15 and RetailRocket. Table 7 shows the results. There is a consistent trend across all three datasets: when objective conflict is strongest, as in Challenge15, MOGCSL achieves the largest gain over PE on the more valuable objective (e.g., around 20% improvement on purchase HR@10), while only slightly sacrificing click performance; when the objectives are more aligned, as in RetailRocket, the gain on purchase remains substantial (e.g., around 17% on purchase HR@10) and the click metrics are nearly the same as PE; and on Tenrec, whose conflict level lies between the other two datasets, MOGCSL remains on par with PE on click while still improving like by about 19% relatively. Taken together, these results suggest that the advantage of MOGCSL becomes more pronounced on the higher-value objective as objective conflict becomes stronger, whereas under weaker conflict it can improve the valuable objective without noticeably harming the secondary one. Overall, compared with strict Pareto-efficiency-based methods such as PE, MOGCSL is better suited to practical recommendation settings where different objectives carry different values.

C.5 Discussion on Effects of Goal Value

Goal selection is central to MOGCSL, since inference performance directly depends on those choosing goals. Our experiments analyze this issue from several perspectives.

First, the value of the inference goal matters. As shown in Section 4.4 and Appendix B.7, sweeping the scaling factor of inference goals reveals a clear non-monotonic pattern: performance is best when the goal is set in a moderate range, while goals that are too small or too large consistently hurt performance. This is consistent with our formulation, where inference selects goals from the achievable set according to the utility principle in Eq.(8) [3] , rather than blindly pursuing arbitrarily large targets. In this sense, MOGCSL is designed to operate within the achievable region and to select reasonable trade-offs among objectives.

Second, the CVAE-based goal-selection module should not be interpreted as choosing a single global goal shared across all examples. Instead, it models a *state-conditioned* goal distribution and selects personalized inference goals for each initial state. For this reason, a conventional grid search over a small set of global goals is not directly comparable to the CVAE-based mechanism. The closest statistical alternative is exactly MOGCSL-S, which sets the inference goal at each timestep to the

[3]Note that there may be multiple samples satisfying this condition. In such cases, we select one with highest purchase.

average cumulative reward in the offline data at the same timestep, scaled by a factor λ chosen by grid search on the validation set. Section 4.4 and Appendix B.6 already compare this statistical strategy with the CVAE-based method in detail. More broadly, our experiments also include comparisons with several grid-search-based baselines, such as MOPRL, Fixed-Weight, and RMTL, and MOGCSL performs favorably against these alternatives.

Third, our current experiments already evaluate the boundary behavior of target goals. The λ-sweep in Appendix B.6 shows that both overly ambitious and overly conservative targets degrade performance. Moreover, the CVAE-based procedure is explicitly designed to avoid arbitrary extrapolation: it samples candidate goals conditioned on the initial state and evaluates them through the estimated distribution of achievable outcomes before selecting one according to the utility function. Therefore, the method is not intended to chase unattainable goals, but to choose desirable goals within the achievable region. Our analysis in Section 4.4 and Appendix B.5 further suggests that when high-goal trajectories are sparse, more optimistic goal-generation can become less effective because the policy lacks sufficient supervision in that region.

Last, we'd like to highlight that MOGCSL is not designed to recover the entire Pareto frontier or to optimize Pareto efficiency as its primary criterion. In recommender systems, different objectives often carry different practical values, so the central question is not whether a method can cover all Pareto-efficient operating points, but whether it can achieve robust and satisfactory performance under application-specific utility preferences. For example, in many recommendation scenarios, a substantial improvement on a higher-value objective such as purchase may be preferable even if it is accompanied by a small reduction on a lower-value objective such as click. Under such settings, evaluating the whole Pareto frontier is not always the most meaningful criterion, because some Pareto-valid regions may correspond to trade-offs that are not practically desirable. In particular, operating points with high click but low purchase may still be Pareto-efficient, yet remain unattractive in realistic recommendation environments. For this reason, our evaluation does not treat Pareto-front recovery as the main goal. Instead, MOGCSL learns the distribution of achievable multi-objective goals and selects operating points from the achievable set according to a customizable utility principle. The resulting analysis therefore focuses on how different achievable-goal selections affect downstream trade-offs and performance, which is more closely aligned with the practical role of goal selection in our application. This experimental design is also consistent with recent work on multi-objective recommendation, such as PMORS [31] and RMTL [30], which likewise do not report full Pareto-front comparisons, even though PMORS uses Pareto-based optimization techniques. Nevertheless, for completeness, we also **provide a Pareto-front comparison between MOGCSL and MMOE-PE** on click and purchase for RetailRocket. Specifically, we vary the utility used in MOGCSL goal selection so as to adjust the relative importance of click and purchase to draw this figure. The results in Figure 4 further demonstrate MOGCSL's advantage in covering high-value areas with strong purchase signals, whereas MMOE-PE's purchase performance is largely constrained by its reliance on click learning.

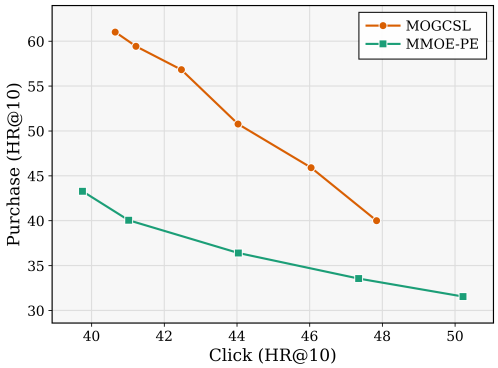

Figure 4: Comparison of Pareto front between MOGCSL and MMOE-PE.

C.6 Discussion on Goal-Selection Strategy

We'd like to further discuss our proposed principled goal-selection strategy in Section 3.3. Specifically, we use two generative models for the two distributional subproblems in Algorithm 2.

First, we use a generative model for the prior $q(g'|s')$. This is necessary because candidate-goal generation is inherently one-to-many: for a new state s', there are multiple feasible trade-offs across objectives, and Algorithm 2 requires multiple candidate goals for comparison.

Table 8: Comparison between regression and CVAE-based strategy for mean estimation.

[RetailRocket]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MOGCSL-R	64.88±0.27	52.76±0.24	69.37±0.30	53.66±0.19	35.98±0.33	24.94±0.18	41.07±0.49	26.21±0.37
MOGCSL-C	65.01±0.07	52.89±0.04	69.34±0.05	54.00±0.04	36.54±0.02	25.41±0.04	42.20±0.03	26.84±0.06

[Challenge15]	Purchase (%)				Click (%)			
	HR@10	NG@10	HR@20	NG@20	HR@10	NG@10	HR@20	NG@20
MOGCSL-R	54.97±0.37	35.07±0.33	63.73±0.25	37.25±0.25	42.03±0.19	25.18±0.33	49.96±0.20	27.29±0.17
MOGCSL-C	55.13±0.07	35.04±0.02	63.98±0.04	37.30±0.03	42.14±0.04	25.37±0.07	50.12±0.09	27.53±0.05

Second, we use a generative model for $P(g_a|s', g', \pi)$ because, under a fixed condition $c = (s', g', \pi)$, there exists only one realized sample from the condition-specific distribution $P(g|c_i)$ in the training data. Since c is high-dimensional and effectively unique per instance, directly regressing to one target treats a single noisy draw as an approximation of the multi-dimensional conditional mean. In contrast, learning the conditional distribution allows $\mathbb{E}[g_a|s', g', \pi]$ to be estimated directly by sampling, which better matches the distributional nature of Algorithm 2.

That said, from a theoretical standpoint, we agree that a regression-based approach could also be applied to approximate the second component. In principle, with unlimited training data, direct regression can converge to the true expected value. However, in Section 3.3, we adopt a generative formulation to remain consistent with the inference procedure. The intuition is that directly learning the conditional distribution itself and estimating its mean by sampling could be more robust with lower variance than directly regressing to the mean from only one noisy realized sample for each condition on limited offline data.

To examine this point, we conduct additional experiments comparing MOGCSL with the CVAE-based strategy against a variant that directly regresses the conditional mean of the achievable goal. Table 8 reports the results. Overall, the CVAE-based version generally performs better, although the performance gap is not large in every case. More importantly, the regression-based variant exhibits substantially higher variance across runs than MOGCSL-C. This supports our intuition that direct regression can provide a noisier estimate of the expectation, whereas the CVAE-based sampling strategy is more robust.

C.7 Discussion on Generalizability and Scalability

From the experiments and discussions in Section 4.4, we observed that different goal-choosing strategies perform differently on datasets with varying reward distributions. However, we emphasize our finding that simpler, statistics-based goal-selection strategies already outperform existing baselines across diverse scenarios, demonstrating MOGCSL's practical applicability especially where lower latency or reduced inference complexity is desired. The advanced CVAE-based algorithm provides additional performance gains specifically on datasets with higher goal values (as shown in Appendix B.8), offering practitioners the flexibility to choose strategies based on dataset characteristics and performance-complexity requirements. As a result, we highlight that this **doesn't weaken the framework's practical value**. Instead, it helps for appropriate strategy selection tailored to practical use cases. As discussed in Section 4.4, this finding is also consistent with recent GCSL theoretical analyses [32], which demonstrates that overly optimistic goal conditioning can be harmful when high-quality trajectories are insufficient.

From another perspective, although the formulation of MOGCSL is presented in a general manner for sequential decision-making problems, the main claim of this paper is that MOGCSL provides a general **supervised framework for multi-objective recommendation**, which is also the motivation and application setting emphasized in the abstract and introduction. We prefer to keep the scope of this work focused on recommender systems and leave broader cross-domain validation to future work. We've included a relevant discussion of further extension and application of MOGCSL into other areas in Appendix D.

For the scalability of MOGCSL, we briefly discussed this in Appendix C.1. Specifically, MOGCSL represents multiple objectives as a single goal vector, so additional objectives can be incorporated without changing the architecture. This design is naturally more scalable than many conventional MTL methods, whose complexity often grows substantially with the number of objectives. Likewise, our goal-selection mechanism is based on the achievable goal set and is naturally extensible to higher-dimensional goals. While additional objectives may require more training data to effectively adapt to higher-dimensional spaces, this challenge is intrinsic to all MTL methods and not unique to ours.

That said, our experimental design follows the standard settings used in prior multi-objective recommendation work, such as PE [11] and PMORS [31], which focus on two-objective experiments. This choice ensures that comparisons remain as direct and fair as possible. In practice, these prior studies are also evaluated only under two-objective recommendation settings, which is why our empirical study adopts the same scope.

At the same time, our goal in this paper is to address the typical and practically important challenges in multi-objective recommendation, rather than to comprehensively study high-dimensional multi-objective learning. To keep this submission focused on recommendation settings, we leave a comprehensive study of higher-dimensional objective spaces to future work.

C.8 Discussion on Generative Models

As illustrated in Section 3.3, we chose CVAE for the goal-selection algorithm primarily because of its simplicity, efficiency, and stable training dynamics, which align closely with our objective of developing scalable methods suitable for real-world recommendation scenarios. Apart from that, we've conducted some preliminary experiments with diffusion-based methods such as DDPO, and the results showed no significant performance improvement over CVAE. Additionally, CVAE demonstrated consistently stable training dynamics and faster inference, essential attributes for practical deployments.

C.9 Discussion on Data Robustness

As illustrated in Section 3.2, MOGCSL assumes the availability of reward signals within sequential data. In recommender systems, such reward signals typically derive naturally and easily from explicit user feedback, such as clicks, purchases, or engagement durations.

However, clear rewards might not be readily available in some other scenarios. For these situations, we propose the following discussions:

- Proxy rewards: Intermediate signals often effectively serve as proxies. For instance, browsing duration, add-to-cart events, or return visits in e-commerce scenarios can reliably approximate user satisfaction when explicit purchase data is sparse.
- Synthetic rewards: In cases like path planning or robotic control, synthetic metrics such as proximity to target states or successful task completion checkpoints can be constructed to substitute explicit rewards.
- Reward inference: Techniques such as inverse reinforcement learning or reward inference from demonstration could integrate seamlessly into MOGCSL, enabling application in less structured environments.

Furthermore, as discussed in Section 3.4, one of the MOGCSL's distinct advantages lies in leveraging long-term goals to effectively handle potentially noisy immediate reward signals. Our experiments in Appendix B.2 demonstrate that this mechanism significantly enhances robustness, enabling effective learning even when datasets contain ambiguous or imprecise signals.

C.10 Discussion on Social Impact

In general, we believe that MOGCSL poses limited privacy and social risk, since it can be trained using standard anonymized user interaction data and does not require collecting additional sensitive information. Moreover, by explicitly modeling multiple objectives, MOGCSL may help recommen-

dation systems better balance different user and platform goals, such as engagement, satisfaction, diversity, and long-term utility.

Nevertheless, as with other recommendation methods, MOGCSL may inherit biases present in the training data. If historical user interactions reflect popularity bias, unfair exposure, or narrow preference patterns, the learned policy may amplify these effects. Therefore, when deploying MOGCSL in real-world systems, it is important to incorporate appropriate safeguards, such as privacy-preserving data processing, fairness-aware evaluation, and monitoring of unintended recommendation behaviors. We leave a more comprehensive study of these social impacts to future work.

D Limitations and Future Works

Although MOGCSL demonstrates strong effectiveness and efficiency for multi-objective recommendation, we identify some potential limitations and corresponding future works.

First, the most prominent gains from the CVAE-based goal-selection strategy are observed on datasets with relatively high average goals. Simpler statistics-based strategies remain effective when average goals are low and offer practitioners flexibility in trading off performance against computational complexity. Nonetheless, further improving the CVAE-based goal-selection mechanism is a valuable direction. Potential approaches include data augmentation techniques (e.g., re-/down-sampling, active learning) to alleviate sparsity in the high-goal region. In addition, extending MOGCSL to an online training paradigm is a promising way to further mitigate this issue if an online environment is accessible: one can pre-train on offline data, deploy the policy to collect new trajectories with higher achieved goals, and iteratively retrain on this enriched dataset.

Second, our current experiments focus on using long-term multi-objective outcomes to downweight noisy instantaneous behaviors that are weak indicators of positive preference. A natural extension is to directly model and leverage explicit negative signals. This could be realized by encoding negative feedback (e.g., quits, skips, dislikes) as additional dimensions in the goal vector, and then incorporating penalties in the utility function to prioritize minimizing those signals during inference.

Third, although our evaluation uses real-world industrial recommendation datasets, we don't conduct online A/B testing with live users due to resource limitations. Industrial deployment and online experimentation would provide additional evidence of practical impact and constitute an important direction, especially from an industry perspective.

Finally, while our study is motivated by empirical challenges in large-scale recommender systems (e.g., efficiency constraints and noisy interaction data), the MOGCSL formulation is broadly applicable to other multi-objective learning tasks. By reframing multi-objective optimization as learning to reach achievable multi-dimensional goals, the framework can potentially be applied to other domains, such as medical prediction, robotic control, and marketing optimization.

E Asset Licenses

Challenge15 [36] is released under the Creative Commons Attribution 4.0 International License (CC BY 4.0). The RetailRocket [37] dataset is available on Kaggle under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0). Tenrec [38] is released under the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0) for the dataset, with accompanying code released under the Apache License 2.0. All datasets are used in compliance with their respective licenses and terms of use.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We proposed a novel framework for multi-objective recommendation within goal-conditional supervised learning framework.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations and future works to address them in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper includes a theoretical analysis to show the property of the distribution of the achievable multi-dimensional goals of GCSL.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We elaborate on the experiment details in the Experiment section, including datasets, baselines, hyper-parameters, etc. We also release the code and data for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code and data are accessible at `https://anonymous.4open.science/r/MOGCSL-D7A2`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We include dataset information and training details in the Experiment part and Appendix. We also clearly describe the knowledge base and formally introduce the task settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For each experiment, the mean and standard deviation over 5 seeds are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We run our experiments on a single NVIDIA RTX 3090 and AMD 3960X.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We do not induce any potential research harm mentioned in NeurIPS Code of Ethics in our paper.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the impact in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our method does not involve such safe issues.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Creators or original owners of assets mentioned in the paper are properly cited and the license and terms of use are respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: This paper does not involve usage of LLMs as component of the core methods.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.