

GOAL-CONDITIONED SUPERVISED LEARNING FOR MULTI-OBJECTIVE RECOMMENDATION

Anonymous authors

Paper under double-blind review

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 more complex optimization problem, particularly when navigating potential conflicts between 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 effectively filters out uninformative or noisy 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 “high” achievable goals for inference.

While MOGCSL is quite general, 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 on real-world recommendation datasets. 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 and data can be found in: <https://anonymous.4open.science/r/MOGCSL-D7A2>.

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 Ruder (2017); Sener & Koltun (2018); Zhang & Yang (2021). 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 Sener & Koltun (2018).

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 previous research focused on designing model architectures specifically for multi-objective scenarios, including the shared-bottom structure Ma et al. (2018), the experts and task gating networks Ma et al. (2018), 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 Liu et al. (2019) or dynamically adjust gradients Yu et al. (2020). We remark that these approaches often introduce substantial space and computational complexity Ma et al. (2018); Zhang & Yang (2021). Also, they treat all the data 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) Yang et al. (2022); Liu et al. (2022) to the multi-objective recommendation problem by introducing a new 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 Chen et al. (2021b); Xin et al. (2022) (e.g., some multiple of the average goal in training). Although we observe that simple choices for MOGCSL do reasonably well in practice, we also 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 general supervised framework called MOGCSL for multi-objective recommendations that, by design, selectively leverages training data with desirable long-term rewards on multiple objectives. 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 novel goal-choosing algorithm that can model the distribution of achievable goals over interaction sequences and choose desirable “high” goals across multiple objectives. This is novel even for the single-objective case and addresses a fundamental challenge in GCSL.
- 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 with multiple objectives, 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 Ma et al. (2018); Misra et al. (2016) and optimization constraints Liu et al. (2019); Lin et al. (2019). All of these works give equal weight to all instances in the training data, 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 Zhang & Yang (2021), making them more challenging for large-scale applications in the real world. For example, MMOE Ma et al. (2018) requires constructing separate towers for each objective. DWA Liu et al. (2019) necessitates recording and calculating loss change dynamics for each training epoch, while PE Lin et al. (2019) demands substantial computational resources to solve an optimization problem for Pareto efficiency. A recent work FAMO Liu et al. (2024) 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 Liu et al. (2022); Chen et al. (2021b); Janner

et al. (2021). 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 Ajay et al. (2020); Zhuang et al. (2024), 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 Cai et al. (2022); Stamenkovic et al. (2022). Also, their evaluation principles are different, typically relying on long-term metrics and synthetic environments. Hence we don't compare with such approaches in this paper.

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 capability of MOGCSL to discount potentially highly noisy samples in the training data.

3.1 A NEW VIEW FOR MULTI-OBJECTIVE LEARNING

Multi-objective learning is typically formulated as an optimization problem over multiple losses Ma et al. (2018); Misra et al. (2016); Yu et al. (2020); Lin et al. (2019), each defined on a distinct objective. Consider a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i^1, y_i^2, \dots, y_i^n)\}_{i \in [1, M]}$, where \mathbf{x}_i represents the feature, y_i^j is the ground-truth score on the j -th objective, and M is the total number of data points. For a given model $f(\mathbf{x}; \theta)$, multiple empirical losses can be computed, one per objective as $\mathcal{L}_j(\theta) = \mathbb{E}_{(\mathbf{x}, y^j) \in \mathcal{D}}[\mathcal{L}(f(\mathbf{x}; \theta), y^j)]$. The model can then be optimized by minimizing a single loss, which is obtained by combining all the losses through a weighted summation as: $\min_{\theta} \sum_{j=1}^n w^j \mathcal{L}_j(\theta)$.

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 Zhang & Yang (2021).

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 an Multi-Objective Markov Decision Process (MOMDP) Roijers et al. (2013). 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 $\tau = \{(s_t, a_t, \mathbf{r}_t)\}_{t \in [1, |\tau|]}$. A state $s_t \in \mathcal{S}$ 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 $\mathbf{r}_t = R(s_t, a_t)$ means the agent receives a reward \mathbf{r}_t after taking an action a_t under state s_t . Note that the reward function $R(s_t, a_t)$ in MOMDP is represented by a multi-dimensional vector instead of a scalar.

In this context, all the objectives can be quantified using reward \mathbf{r}_t at each time step. Specifically, since \mathbf{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 \mathbf{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 from the current timestep to the end of the trajectory:

$$\mathbf{g}_t = \sum_{t'=t}^{|\tau|} \mathbf{r}_{t'}, \quad (1)$$

Algorithm 1: Training of MOGCSL

```

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

```

where g_t can be called as a ‘‘goal’’ in the literature of GCSL Yang et al. (2022).

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 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 $\arg \max_\theta \mathbb{E}_{\mathcal{D}_{tr}}[\log \pi_\theta(a|s, g)]$. 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 (s_t, a_t, r_t) with (s_t, a_t, g_t) , where g_t is defined according to Eq. (1). Subsequently, we employ a sequential model Kang & McAuley (2018) 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 Kang & McAuley (2018). Specifically, let the interaction history of a user up to time t be denoted as $v_{1:t-1} = \{v_1, \dots, v_{t-1}\}$. We first map each item $v \in \mathcal{V}$ into the embedding space, resulting in the embedding representation of the history: $e_{1:t-1} = [e_1, \dots, e_{t-1}]$. Then we encode $e_{1:t-1}$ 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_{g_t} through a simple fully connected (FC) layer with a subsequent normalization module. The final representation of state s_t is derived by concatenating the sequential encoding, timestep embedding and goal embedding together:

$$emb_{s_t} = \text{Concat}(T\text{-enc}(e_{1:t-1}), emb_t, emb_{g_t}). \quad (2)$$

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

$$\mathbf{M}_{s_t, g_t} = \text{Atten}(emb_{g_t}). \quad (3)$$

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):

$$[\pi_\theta(v^1|s_t, g_t), \dots, \pi_\theta(v^N|s_t, g_t)] = \delta(\text{MLP}(\mathbf{M}_{s_t, g_t})), \quad (4)$$

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:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t, a_t, g_t) \in \mathcal{D}_{tr}}[-\log(\pi_\theta(a_t|s_t, g_t))]. \quad (5)$$

Algorithm 2: Inference of MOGCSL

-
- 1 **Input:** state s' , sample size K , policy model π , utility principle U , prior $q(\mathbf{g}'|s')$, distribution of achievable goals $P(\mathbf{g}^a|s', \mathbf{g}', \pi)$
 - 2 **Initialization:** set of potential input goals $\mathbf{G}' = \emptyset$, set of expected achievable goals $\mathbf{G}^a = \emptyset$
 - 3 **for** $k = 1, \dots, K$ **do**
 - 4 Sample a \mathbf{g}'_k from $q(\mathbf{g}'|s')$
 - 5 Compute the expectation of the achievable goal through sampling: $\tilde{\mathbf{g}}_k^a = \mathbb{E}_{\mathbf{g}^a \sim P(\cdot|s', \mathbf{g}'_k, \pi)} \mathbf{g}^a$
 - 6 $\mathbf{G}' = \mathbf{G}' \cup \mathbf{g}'_k$
 - 7 $\mathbf{G}^a = \mathbf{G}^a \cup \tilde{\mathbf{g}}_k^a$
 - 8 Choose the best $\tilde{\mathbf{g}}_b^a$ from \mathbf{G}^a according to $U(\tilde{\mathbf{g}}^a)$
 - 9 Choose corresponding \mathbf{g}'_b from \mathbf{G}'
 - 10 **Return:** $\pi(\cdot|s', \mathbf{g}'_b)$
-

The training process is illustrated in Algorithm 1.

3.3 MOGCSL INFERENCE

After training, we derive a model $\pi_\theta(a|s, \mathbf{g})$ 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 via Eq. (1), we must assign a desirable goal as input for the new state encountered during inference. GCSL approaches typically determine this goal-choosing strategy based on simple statistics calculated from the training data. E.g., Chen et al. (2021b) and Zheng et al. (2022) 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. (2022) 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 $\pi(a|s, \mathbf{g})$ given the initial state \mathbf{s}_1 and goal \mathbf{g}_1 , the distribution of goals \mathbf{g}^a (i.e., cumulative rewards) that the agent actually achieves throughout the trajectory is determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi)$.*

Based on this theorem, we'd like to learn the distribution of \mathbf{g}^a conditioned on $(\mathbf{s}_1, \mathbf{g}_1, \pi)$, denoted as $P(\mathbf{g}^a|\mathbf{s}_1, \mathbf{g}_1, \pi)$. General generative models, such as GANs Mirza & Osindero (2014) and diffusion models Ho & Salimans (2022), can be employed to learn this distribution. In this paper, we propose the use of a conditional variational auto-encoder (CVAE) Sohn et al. (2015) 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 $(\mathbf{s}, \mathbf{g}) \in \mathcal{D}_{tr}$, \mathbf{g} should be a sample from the distribution of the achievable goals by the policy π , given the initial state \mathbf{s} and input goal \mathbf{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 (\mathbf{s}, \mathbf{g}) is exactly \mathbf{g} . Let $\mathbf{c} = (\mathbf{s}, \mathbf{g}, \pi)$. The loss function is:

$$\mathcal{L}_{CVAE1} = \mathbb{E}_{(\mathbf{s}, \mathbf{g}) \in \mathcal{D}_{tr}, z \sim Q_1} [\log P_1(\mathbf{g}|z, \mathbf{c}) + D_{KL}(Q_1(z|\mathbf{g}, \mathbf{c})||P(z))], \quad (6)$$

where $Q_1(z|\mathbf{g}, \mathbf{c})$ is the encoder and $P_1(\mathbf{g}|z, \mathbf{c})$ is the decoder. Based on Gaussian distribution assumption, they can be written as $Q_1 = \mathcal{N}(\mu(\mathbf{g}, \mathbf{c}), \Sigma(\mathbf{g}, \mathbf{c}))$ and $P_1 = \mathcal{N}(f_{CVAE1}(z, \mathbf{c}), \sigma^2 I)$ respectively, where $z \sim \mathcal{N}(0, I)$. Then we can derive a sample of \mathbf{g}^a by inputting a sampled z into f_{CVAE1} .

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

$$\mathcal{L}_{CVAE2} = \mathbb{E}_{(\mathbf{s}, \mathbf{g}) \in \mathcal{D}_{tr}, z \sim Q_2} [\log P_2(\mathbf{g}|z, \mathbf{s}) + D_{KL}(Q_2(z|\mathbf{g}, \mathbf{s})||P(z))]. \quad (7)$$

270 Finally, we’ll choose a desirable goal as the input along with the new state s' encountered in inference
 271 by sampling from the two CVAE models. Specifically, we propose to: (1) sample from the prior
 272 $q(g'|s')$ to get a set of potential input goals, denoted as G' , (2) for each $g' \in G'$, estimate the
 273 expectation of the actually achievable goal \tilde{g}^a by sampling from $P(\cdot|s', g', \pi)$ and taking the average,
 274 (3) choose a best goal as input for inference from G' according to the associated expected \tilde{g}^a by a
 275 predefined utility principle $U(\tilde{g}^a)$, which generally tends to pick up a “high” goal to achieve larger
 276 rewards on multiple objectives. The exact definition can be customized based on specific business
 277 requirements or objective priorities. E.g., choose by a predefined partial ordering¹. Note that our
 278 algorithm is general and can accommodate adaptations of the utility function, as the desired goals are
 279 always set among the set of achievable goals, which naturally resolves the objective-conflicting issue.
 280 This definition flexibility allows practitioners to easily tailor MOGCSL to various practical scenarios
 281 and objective priorities. See Algorithm 2 for detailed inference pseudocode.

283 3.4 ANALYSIS OF DENOISING CAPABILITY

284
 285 An important benefit of MOGCSL is its capability to remove harmful effects of potentially noisy
 286 instances in the training data by leveraging the multiple-objective goals. To illustrate this, we consider
 287 the following setup that is common in recommender systems. There is a recommender system that
 288 has been operational, and recording data. At each interaction, the system shows the user a short
 289 list of items. The user then chooses one of these items. In the counterfactual that the recommender
 290 system is ideal, the action recorded would be a which reveals the user’s true interest. Since the actual
 291 recommender system to collect the data is not ideal, we have no direct access to a , but rather to a
 292 noisy version of it $\varepsilon(a)$.

293 We assume that the noisy portion of the training data originates from users who are presented with
 294 a list of items that are not suitable for them, rendering their reactions to these recommendations
 295 uninformative. Conversely, interactions achieving higher goals are generally less noisy, meaning
 296 $\varepsilon(a)$ is closer to a . To illustrate this, consider a scenario where a user clicks two recommended
 297 items (v_1 and v_2) under the same state. After clicking v_1 , the user chooses to quit the system,
 298 while he stays longer and browses more items after clicking v_2 . This indicates that the goal (i.e.,
 299 cumulative reward) with v_2 is larger than that with v_1 . In this case, we argue that v_2 should be
 300 considered as the user’s truly preferred item over v_1 . That’s because the act of quitting, which results
 301 in a smaller goal, indicates user dissatisfaction with the previous recommendation, even though
 302 he did click v_1 before. Our proposed MOGCSL can model and leverage this mechanism based on
 303 multi-dimensional goals, which serve as a description of the **future effects of current actions on**
 304 **multiple objectives**. Specifically, by incorporating multi-dimensional goals as input, MOGCSL can
 305 effectively differentiate between noisy and noiseless samples in the training data. During inference,
 306 when high goals are specified as input, the model can make predictions based primarily on the patterns
 307 learned from the corresponding noiseless interaction data. *It’s worth noting that several prior studies*
 308 *on recommender system denoising Zhang et al. (2025); Chen et al. (2021a) have highlighted that*
 309 *such noises (i.e., users’ noisy instant behaviors that don’t align with their long-term, real interests)*
 310 *are very common and widely exist in real-world recommendation datasets.*

311 To empirically validate this effect, we conduct experiments that are illustrated in Appendix B.2 due
 312 to limited space. The results demonstrate the denoising capability of MOGCSL.

314 4 EXPERIMENTS

315
 316 In this section, we introduce our experiments on two e-commerce datasets, aiming to address the
 317 following research questions: 1) **RQ1**. How does MOGCSL perform when compared to previous
 318 methods for multi-objective learning in recommender systems? 2) **RQ2**. How does MOGCSL
 319 mitigate the complexity challenges, including space and time complexity, as well as the intricacies of
 320 parameter tuning encountered in prior research? 3) **RQ3**. How does the goal-generation module for
 321 inference perform when compared to strategies based on simple statistics?
 322
 323

¹See Section 4.4 for the implementation in our experiments.

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. More details of the datasets, metrics, and implementation specifics are provided in Appendix B.

Baselines Prior research on multi-objective learning encompass both model structure adaptation and optimization constraints. In our experiments, we consider two representative model architectures: Shared-Bottom Ma et al. (2018) and MMOE Ma et al. (2018). For works on optimization constraints, we compare four methods: Fixed-Weights Wang et al. (2016) assigns fixed weights for different objectives based on grid search; DWA Liu et al. (2019) dynamically adjusts weights by considering the dynamics of loss change; PE Lin et al. (2019) generates Pareto-efficient recommendations across multiple objectives; [Nash-MTL Navon et al. \(2022\) characterize the Nash bargaining solution for multi-objective learning](#); FAMO Liu et al. (2024) adjusts weights to achieve balanced task loss reduction while maintaining relatively low space and time complexity.

Following previous research Yu et al. (2020), we consider all these optimization methods for each model architecture, resulting in eight baselines denoted as Share-Fix, Share-DWA, Share-PE, Share-Nash-MTL, Share-FAMO, MMOE-Fix, MMOE-DWA, MMOE-PE, MMOE-Nash-MTL, MMOE-FAMO. Additionally, we introduce a variant of a recent work called PRL Xin et al. (2022), 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 Liu et al. (2023), 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 Jin et al. (2024), a recent supervised multi-objective learning framework designed for recommender systems. [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.](#)

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 ($NDCG@k$). We use the abbreviation NG to denote NDCG in the tables. For each experiment, the mean and standard deviation over 5 seeds are reported.

4.2 PERFORMANCE COMPARISON (RQ1)

We begin by conducting experiments to compare the performance of MOGCSL and selected baselines in terms of top- k recommendation. The experimental results are presented 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 Xin et al. (2022). Specifically, at each timestep in inference, the goal 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 and meaningful 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 benchmarks in terms of purchase-related metrics. 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 structure in many cases. Surprisingly, a naive optimization strategy based on fixed loss weights can outperform more advanced methods like DWA across several metrics (e.g., Share-Fix vs Share-DWA). These findings highlight the limitations of previous approaches that rely on assumptions about model architectures or optimization constraints, which may not be necessarily true in general environments. Similar trends are observed on Challenge15. While MMOE-PE performs slightly better on click metrics 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.

Table 1: Comparison between MOGCSL and other baselines on RetailRocket and Challenge15 datasets. The mean and standard deviation over 5 seeds are reported. Boldface denotes the best results.

[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±0.55	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

However, this assumption does not align well with the realities of recommender systems. In practice, a significant improvement in high-value objectives (such as purchase) accompanied by only minor reductions in less critical objectives (such as clicks) is often preferable in practical scenarios.

Apart from previous 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 $r^t = w_c r^c + w_p r^p$, 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 show that MOGCSL consistently outperforms MOPRL across all weight combinations on both click and purchase metrics, demonstrating that representing the goal as a multi-dimensional vector enhances the effectiveness of GCSL on multi-objective learning. See the figure for the comparison in Appendix B.5.

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 Zhang & Yang (2021). For instance, MMOE and Shared-Bottom both design separate towers for each objective Ma et al. (2018), 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. MOGCSL, in the other hand, 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 $O(m^n)$ 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: Comparison of time and space complexity 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-MTL	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-MTL	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

Table 2 summarizes the complexity comparison. It’s evident that MOGCSL significantly benefits from a smaller model size and faster training speed, while concurrently achieving great performance.

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 $U(\mathbf{g})$ 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 $\tilde{\mathbf{g}}_b^a$ from the set \mathbf{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:

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

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 (Algorithm 2). 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 slightly 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. Interestingly, similar findings have been reported in prior research Chen et al. (2021b); Xin et al. (2022); Zheng et al. (2022), 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 a dataset with higher average goals. Within our MOGCSL framework, we define click and like as the two dimensions of the multi-objective goals.

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 0.55	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	42.27 \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

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.

Table 3 shows the comparison results, where MOGCSL-C significantly outperforms MOGCSL-S. Together with previous experiments on datasets featuring lower average goals, these results reinforce our conclusion and provide clear practical insights into selecting between simple and advanced goal-selection strategies based on dataset properties. [It’s worth noting that a very recent work, RVDT Bai et al. \(2025\), 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.](#)

The results provide several insights for selecting goal-choosing strategies when applying MOGCSL in practical applications. First, strategies based on simple statistics on the training data prove to be efficient and effective in many cases, particularly when low latency or reduced model 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, it’s worth noting that [our core technical contribution lies in developing a novel multi-objective learning framework leveraging GCSL, addressing and modeling the achievability of 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 valuable and non-trivial finding.](#)

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. Building upon GCSL, 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. For future work, we aim to explore a more effective goal-generation strategy for inference, which may necessitate a change in the training paradigm.

540 REPRODUCIBILITY STATEMENT

541 We provide anonymized code at <https://anonymous.4open.science/r/MOGCSL-D7A2>, together with
542 the datasets used in our experiments. Comprehensive details of the experimental setup, including
543 hyperparameters, training details, and evaluation methods, are presented in the Experiments section
544 and Appendix. With these resources, we are confident that readers will be able to reproduce the
545 results presented in the paper.
546

547 REFERENCES

- 548 Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive
549 discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.
- 550 Wensong Bai, Chufan Chen, Yichao Fu, Qihang Xu, Chao Zhang, and Hui Qian. Rebalancing return
551 coverage for conditional sequence modeling in offline reinforcement learning. In *The Thirty-ninth
552 Annual Conference on Neural Information Processing Systems, 2025*.
- 553 Qingpeng Cai, Ruohan Zhan, Chi Zhang, Jie Zheng, Guangwei Ding, Pinghua Gong, Dong Zheng,
554 and Peng Jiang. Constrained reinforcement learning for short video recommendation. *arXiv
555 preprint arXiv:2205.13248*, 2022.
- 556 Jiawei Chen, Xiang Wang, Fuli Feng, and Xiangnan He. Bias issues and solutions in recommender
557 system: Tutorial on the recsys 2021. In *Proceedings of the 15th ACM conference on recommender
558 systems*, pp. 825–827, 2021a.
- 559 Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel,
560 Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence
561 modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021b.
- 562 Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the
563 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794,
564 2016.
- 565 Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based
566 recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.
- 567 Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*,
568 2022.
- 569 Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence
570 modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.
- 571 Jipeng Jin, Zhaoxiang Zhang, Zhiheng Li, Xiaofeng Gao, Xiongwen Yang, Lei Xiao, and Jie Jiang.
572 Pareto-based multi-objective recommender system with forgetting curve. In *Proceedings of the
573 33rd ACM International Conference on Information and Knowledge Management*, pp. 4603–4611,
574 2024.
- 575 Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE
576 international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.
- 577 Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang,
578 Wenwu Ou, and Peng Jiang. A pareto-efficient algorithm for multiple objective optimization in
579 e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender
580 systems*, pp. 20–28, 2019.
- 581 Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization.
582 *Advances in Neural Information Processing Systems*, 36, 2024.
- 583 Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems
584 and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

- 594 Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention.
595 In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp.
596 1871–1880, 2019.
- 597 Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen,
598 Tonghao He, Dong Zheng, Peng Jiang, et al. Multi-task recommendations with reinforcement
599 learning. In *Proceedings of the ACM web conference 2023*, pp. 1273–1282, 2023.
- 601 Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships
602 in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD*
603 *international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.
- 604 Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*
605 *arXiv:1411.1784*, 2014.
- 607 Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for
608 multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern*
609 *recognition*, pp. 3994–4003, 2016.
- 611 Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and
612 Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.
- 613 Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-
614 objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113,
615 2013.
- 616 Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint*
617 *arXiv:1706.05098*, 2017.
- 619 Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in*
620 *neural information processing systems*, 31, 2018.
- 622 Kihyuk Sohn, Honglak Lee, and Xinchun Yan. Learning structured output representation using deep
623 conditional generative models. *Advances in neural information processing systems*, 28, 2015.
- 624 Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas.
625 Choosing the best of both worlds: Diverse and novel recommendations through multi-objective
626 reinforcement learning. In *Proceedings of the Fifteenth ACM International Conference on Web*
627 *Search and Data Mining*, pp. 957–965, 2022.
- 628 Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for
629 long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.
- 631 Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Self-supervised reinforce-
632 ment learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR*
633 *conference on research and development in Information Retrieval*, pp. 931–940, 2020.
- 634 Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and
635 Zhaochun Ren. Rethinking reinforcement learning for recommendation: A prompt perspective. In
636 *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in*
637 *Information Retrieval*, pp. 1347–1357, 2022.
- 639 Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie
640 Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv*
641 *preprint arXiv:2202.04478*, 2022.
- 642 Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn.
643 Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:
644 5824–5836, 2020.
- 645 Kaike Zhang, Qi Cao, Fei Sun, Yunfan Wu, Shuchang Tao, Huawei Shen, and Xueqi Cheng. Robust
646 recommender system: a survey and future directions. *ACM Computing Surveys*, 58(1):1–38, 2025.

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

650

651 Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international*
652 *conference on machine learning*, pp. 27042–27059. PMLR, 2022.

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

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702 A PROOF OF THEOREM 1

703 We begin by first proving the following lemma.

704 **Lemma 1.** Assume that the environment is modeled as an MOMDP. Consider a trajectory τ that is
705 generated by the policy $\pi(a|\mathbf{s}, \mathbf{g})$ given the initial state \mathbf{s}_1 and goal \mathbf{g}_1 , the distributions of \mathbf{s}_t and
706 $(\mathbf{s}_t, \mathbf{g}_t)$ at each timestep are both determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a|\mathbf{s}, \mathbf{g}))$.

707 *Proof.* First, for $t = 1$ we have:

$$708 \Pr(\mathbf{r}_1 = \mathbf{r}) = \sum_a \Pr(\mathbf{r}|\mathbf{s}_1, a) = \mathbf{r})\pi(a|\mathbf{s}_1, \mathbf{g}_1). \quad (9)$$

709 Note that the reward function $\mathbf{R}(\mathbf{s}, a)$ is fixed for the given environment. Then, we complete the
710 proof by mathematical induction.

711 **Statement:** The distributions of \mathbf{s}_t and $(\mathbf{s}_t, \mathbf{g}_t)$ are both determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a|\mathbf{s}, \mathbf{g}))$, for
712 $t = 2, 3, \dots, |\tau|$.

713 **Base case $t = 2$:** Since \mathbf{s}_1 and \mathbf{g}_1 are given and fixed, we have:

$$714 \Pr(\mathbf{s}_2 = \mathbf{s}) = \sum_a T(\mathbf{s}|a, \mathbf{s}_1)\pi(a|\mathbf{s}_1, \mathbf{g}_1). \quad (10)$$

715 It's clear that $\mathbf{s}_2 \sim f_{\mathbf{s}_2}(\mathbf{s}; \mathbf{s}_1, \mathbf{g}_1, \pi)$ where $f_{\mathbf{s}_2}$ is a distribution determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi)$.

716 For $(\mathbf{s}_2, \mathbf{g}_2)$, according to the definition of \mathbf{g}_t in Eq. (1), when a reward r_t is received, the desired goal
717 on next timestep is $\mathbf{g}_{t+1} = \mathbf{g}_t - r_t$. Combined with Eq. (9), We have:

$$718 \Pr(\mathbf{s}_2 = \mathbf{s}, \mathbf{g}_2 = \mathbf{g}) = \sum_a \Pr(\mathbf{g}_1 - \mathbf{g}|\mathbf{s}_1, a) = \mathbf{g}_1 - \mathbf{g})T(\mathbf{s}|\mathbf{s}_1, a)\pi(a|\mathbf{s}_1, \mathbf{g}_1). \quad (11)$$

719 Since the dynamic function $T(\mathbf{s}'|\mathbf{s}, a)$ is given, it's clear that $(\mathbf{s}_2, \mathbf{g}_2) \sim f_{\mathbf{s}_2, \mathbf{g}_2}(\mathbf{s}, \mathbf{g}; \mathbf{s}_1, \mathbf{g}_1, \pi)$.

720 **Inductive Hypothesis:** Suppose the statement holds for all t up to some n , $2 \leq n \leq |\tau| - 1$.

721 **Inductive Step:** Let $t = n + 1$, similar to the base case, we have:

$$722 \Pr(\mathbf{s}_{n+1} = \mathbf{s}) = \sum_{a', \mathbf{s}', \mathbf{g}'} T(\mathbf{s}|\mathbf{s}', a')\pi(a'|\mathbf{s}', \mathbf{g}')\Pr(\mathbf{s}_n = \mathbf{s}', \mathbf{g}_n = \mathbf{g}'). \quad (12)$$

$$723 \Pr(\mathbf{s}_{n+1} = \mathbf{s}, \mathbf{g}_{n+1} = \mathbf{g}) = \sum_{a', \mathbf{s}', \mathbf{g}'} [\Pr(\mathbf{r}' = \mathbf{g}' - \mathbf{g}|\mathbf{s}', a')T(\mathbf{s}|\mathbf{s}', a') \\ 724 \cdot \pi(a'|\mathbf{s}', \mathbf{g}')\Pr(\mathbf{s}_n = \mathbf{s}', \mathbf{g}_n = \mathbf{g}')]. \quad (13)$$

725 According to the hypothesis that the distributions of $(\mathbf{s}_n, \mathbf{g}_n)$ is determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a|\mathbf{s}, \mathbf{g}))$,
726 it's easy to see that $\mathbf{s}_{n+1} \sim f_{\mathbf{s}_{n+1}}(\mathbf{s}; \mathbf{s}_1, \mathbf{g}_1, \pi)$ and $(\mathbf{s}_{n+1}, \mathbf{g}_{n+1}) \sim f_{\mathbf{s}_{n+1}, \mathbf{g}_{n+1}}(\mathbf{s}, \mathbf{g}; \mathbf{s}_1, \mathbf{g}_1, \pi)$.

727 As a result, the statement holds for $t = n + 1$. By the principle of mathematical induction, the
728 statement holds for all $t = 2, 3, \dots, |\tau|$. Apparently, that proves Lemma 1. \square

729 Then, based on the lemma, we can prove Theorem 1.

730 *Proof.* Let $|\tau| = T$, by definition we have:

$$731 \mathbf{g}^a = \sum_{t=1}^T \mathbf{r}_t, \quad (14)$$

732 Let $\mathbf{x}_n = (\mathbf{r}_n, \dots, \mathbf{r}_1)$, according to the Markov property and Bayes' rules we have:

$$733 P(\mathbf{r}_n|\mathbf{r}_{n-1}, \dots, \mathbf{r}_1) = P(\mathbf{r}_n|\mathbf{x}_{n-1}) \\ 734 = \sum_{\mathbf{s}_n} P(\mathbf{r}_n|\mathbf{s}_n, \mathbf{x}_{n-1})P(\mathbf{s}_n|\mathbf{x}_{n-1}) \\ 735 = \sum_{\mathbf{s}_n} P(\mathbf{r}_n|\mathbf{s}_n) \sum_{\mathbf{s}_{n-1}} P(\mathbf{s}_n|\mathbf{s}_{n-1}, \mathbf{x}_{n-1})P(\mathbf{s}_{n-1}|\mathbf{x}_{n-1}) \\ 736 = \sum_{\mathbf{s}_n} P(\mathbf{r}_n|\mathbf{s}_n) \sum_{\mathbf{s}_{n-1}} P(\mathbf{s}_n|\mathbf{s}_{n-1}, \mathbf{x}_{n-1}) \dots \sum_{\mathbf{s}_2} P(\mathbf{s}_3|\mathbf{s}_2, \mathbf{x}_{n-1})P(\mathbf{s}_2|\mathbf{x}_{n-1}) \quad (15)$$

For the first term, we have:

$$P(\mathbf{r}_n | \mathbf{s}_n) = \sum_{a_n, \mathbf{g}_n} \frac{\Pr(\mathbf{r}_n = \mathbf{s}_n, a_n) \pi(a_n | \mathbf{s}_n, \mathbf{g}_n) P(\mathbf{s}_n, \mathbf{g}_n)}{P(\mathbf{s}_n)}. \quad (16)$$

Since $(\mathbf{s}_1, \mathbf{g}_1)$ is given and fixed, for each $m \in [2, n-1]$ we have:

$$P(\mathbf{s}_{m+1} | \mathbf{s}_m, \mathbf{x}_{n-1}) = \sum_{a_m} \pi(a_m | \mathbf{s}_m, \mathbf{g}_1 - \sum_{i=1}^{m-1} \mathbf{r}_i) T(\mathbf{s}_{m+1} | \mathbf{s}_m, a_m) \Pr(\mathbf{r}_m | \mathbf{s}_m, a_m). \quad (17)$$

Similarly, the last term can be written as:

$$P(\mathbf{s}_2 | \mathbf{x}_{n-1}) = \sum_{a_1} \pi(a_1 | \mathbf{s}_1, \mathbf{g}_1) T(\mathbf{s}_2 | \mathbf{s}_1, a_1) \Pr(\mathbf{r}_1 | \mathbf{s}_1, a_1). \quad (18)$$

According to Lemma 1, the distributions of \mathbf{s}_n and $(\mathbf{s}_n, \mathbf{g}_n)$ are both determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a | \mathbf{s}, \mathbf{g}))$. As a result, by Eq. (15 - 18), it's clear that the distribution of the conditional probability distribution $P(\mathbf{r}_{n+1} | \mathbf{r}_n, \dots, \mathbf{r}_1)$ is also determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a | \mathbf{s}, \mathbf{g}))$. Then, the distribution of \mathbf{g}^a can be written as:

$$\begin{aligned} \Pr(\mathbf{g}^a = \mathbf{g}) &= \int \cdots \int_{\sum_{i=1}^T \mathbf{r}_i = \mathbf{g}} f(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T) d\mathbf{r}_1 d\mathbf{r}_2 \dots d\mathbf{r}_T \\ &= \int \cdots \int_{\sum_{i=1}^T \mathbf{r}_i = \mathbf{g}} f_1(\mathbf{r}_1) f_2(\mathbf{r}_2 | \mathbf{r}_1) \dots f_T(\mathbf{r}_T | \mathbf{r}_{T-1}, \dots, \mathbf{r}_1) d\mathbf{r}_1 \dots d\mathbf{r}_T \end{aligned} \quad (19)$$

Obviously, the distribution of \mathbf{g}^a is determined by $(\mathbf{s}_1, \mathbf{g}_1, \pi(a | \mathbf{s}, \mathbf{g}))$, which is exactly what Theorem 1 states.

□

B EXPERIMENT DETAILS

B.1 MODEL STRUCTURE

The model structure of MOGCSL is shown in Figure 1.

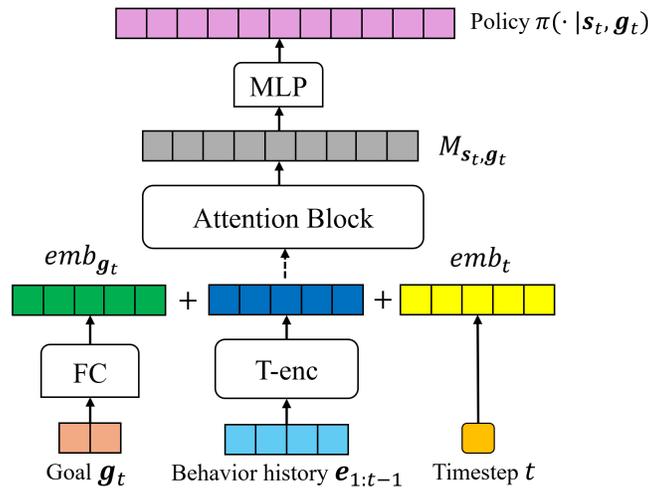


Figure 1: Model structure of MOGCSL.

Table 4: 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

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) $\mathbf{g} = (g_1, \dots, g_n)$ 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 $(\mathbf{s}, \mathbf{g}, \varepsilon(a))$, where the state of the user is represented by a vector \mathbf{s} .

To empirically show the effect of this phenomenon in a simple set-up, we generate a dataset as follows²: (1) The states and the goals are sampled from two independent multivariate normal distributions with dimension of 50 and 5 respectively. (2) The ground-truth action a is entirely determined by \mathbf{s} , whose ID is set to the number of coordinates of \mathbf{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 \mathbf{g} and \mathbf{s} as follows:

$$\varepsilon(a) = \left(\prod_{i=0}^n \mathbb{1}(g_i > -1)\right)a + \left(1 - \prod_{i=0}^n \mathbb{1}(g_i > -1)\right)\text{randint}[1, N], \quad (20)$$

where $a = \sum_{j=0}^l \mathbb{1}(s_j > 0)$.

Since MOGCSL is applicable to any supervised model by integrating goals as additional input features, we choose a simple XGBoost classifier Chen & Guestrin (2016) 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 \mathbf{s} as input and ignores \mathbf{g} . It cannot detect noisy instances in \mathcal{D} because it lacks access to \mathbf{g} . (2) XGBoost-ug: this variant is taken as a single-objective GCSL model, which takes \mathbf{s} and only the first coordinate g_1 of \mathbf{g} as input. Clearly, it’s also unable to precisely distinguish noisy data since $\varepsilon(a)$ is determined by all dimensions of \mathbf{g} (as shown in Eq. (20)). (3) XGBoost-mg: this variant is based on our multi-objective GCSL, which takes both \mathbf{s} and \mathbf{g} as input. It is the only one capable of distinguishing all the noisy data by learning the determination pattern from \mathbf{g} and \mathbf{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 \mathbf{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 4. 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.

²We generate synthetic data due to the unavailability of labels distinguishing between random and genuinely preference-driven user interactions in real-world datasets.

864 B.2.1 DATASETS

865 We conduct experiments on two publicly available datasets: Challenge15³ and RetailRocket⁴. They
866 are both collected from online e-business platforms by recording users' sequential behaviours in
867 recommendation sessions. Specifically, both datasets include binary labels indicating whether a user
868 clicked or purchased the currently recommended item. Following previous research Xin et al. (2022;
869 2020), we filter out sessions with lengths shorter than 3 or longer than 50 to ensure data quality.
870

871 After preprocessing, the Challenge15 dataset comprises 200,000 sessions, encompassing 26,702
872 unique items, 1,110,965 clicks and 43,946 purchases. Similarly, the processed RetailRocket dataset
873 consists of 195,523 sessions, involving 70,852 distinct items. It documents 1,176,680 clicks and
874 57,269 purchases. We partition them into training, validation, and test sets, maintaining an 8:1:1
875 ratio.

876 B.2.2 BASELINE DETAILS

877 In the experiments, we compare two representative model architectures for multi-objective learning:
878

- 879 • **Shared-Bottom** Ma et al. (2018): A classic model structure for multi-objective learning.
880 The bottom of the model is a neural network shared across all objectives. On top of this
881 shared base, separate towers are added for each objective, producing predictions specific to
882 that objective.
- 883 • **MMOE** Ma et al. (2018): A widely used multi-objective model architecture. It first maps
884 inputs to multiple expert modules shared by all objectives. These experts contribute to each
885 objective through designed gates. The final input for each tower is a weighted summation of
886 the experts' outputs.
887

888 Beyond architectural adaptations, other works focus on studying optimization constraints, mainly
889 through adjusting weights of losses for different objectives. We compare the following methods:
890

- 891 • **Fixed-Weights** Wang et al. (2016): A straightforward strategy that assigns fixed weights
892 based on grid search results from the validation set. These weights remain constant through-
893 out the whole training stage.
- 894 • **DWA** Liu et al. (2019): This method aims to dynamically assign weights by considering the
895 rate of loss change for each objective during recent training epochs. Generally, it tends to
896 assign larger weights to objectives with slower loss changes.
- 897 • **PE** Lin et al. (2019): It's designed for generating Pareto-efficient recommendations across
898 multiple objectives. The model optimizes for Pareto efficiency, ensuring no further improve-
899 ment in one objective comes at the expense of any others.
- 900 • **Nash-MTL** Navon et al. (2022): This method frames the gradient combination step in
901 multi-objective learning as a bargaining game and use the Nash bargaining solution to find
902 the optimal update direction.
- 903 • **FAMO** Liu et al. (2024): This recent method aims to dynamically adjust the weights for
904 multi-objective learning, achieving balanced task loss reduction while maintaining relatively
905 low space and time complexity.
906

907 Note that to ensure a fair comparison, we employ the *T-enc* and self-attention block introduced in
908 Section 3.2 as the base module to encode sequential data for all compared baselines.
909

910 B.3 EVALUATION METRICS

911 We employ two widely recognized information retrieval metrics to evaluate model performance in
912 top-*k* recommendation. Hit Ratio ($HR@k$) is to quantify the proportion of recommendations where
913 the ground-truth item appears in the top-*k* positions of the recommendation list Hidasi et al. (2015).
914 Normalized discounted cumulative gain ($NDCG@k$) further considers the positional relevance of
915

916 ³<https://recsys.acm.org/recsys15/challenge>

917 ⁴<https://www.kaggle.com/retailrocket/ecommerce-dataset>

ranked items, assigning greater importance to top positions during calculation Kang & McAuley (2018). 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. 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.

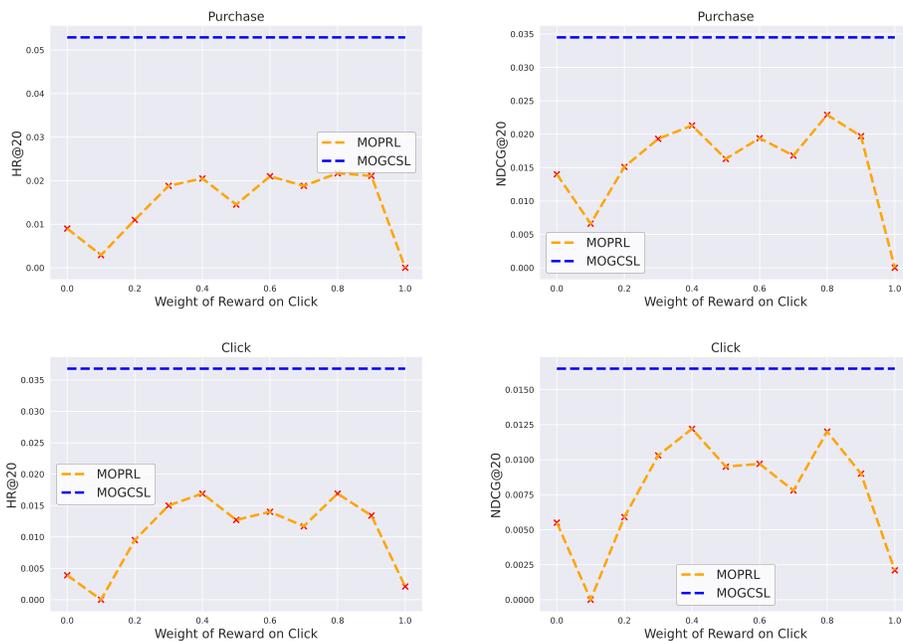


Figure 2: Comparison between MOGCSL and MOPRLs with different weight combinations on RetailRocket. Performance of MOGCSL is not dependent on the weights.

B.6 COMPARISON OF MOGCSL-S W.R.T FACTORS

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

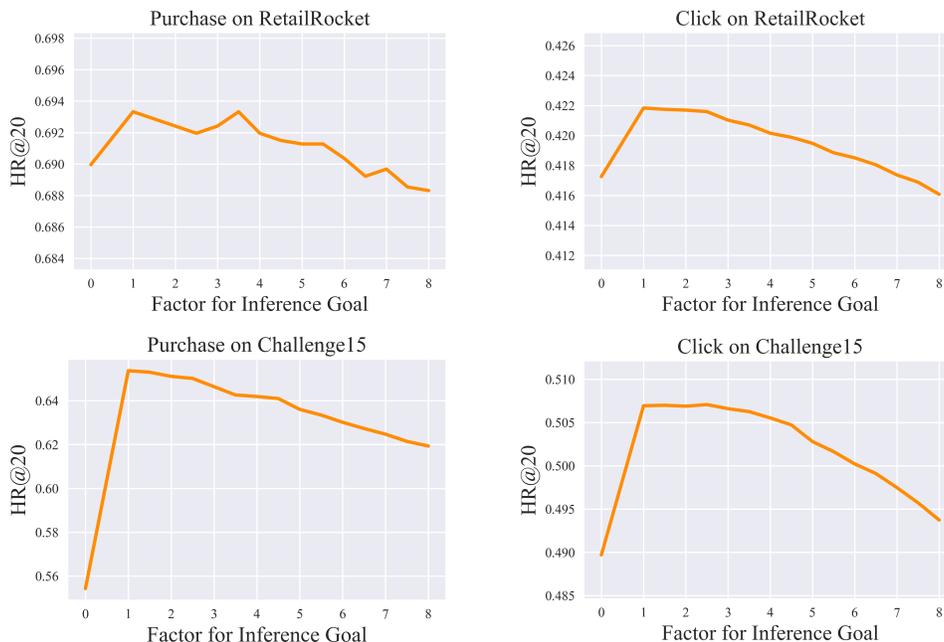


Figure 3: Performance of MOGCSL-S with different factors for inference goal.

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 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 5, 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

Table 5: 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-MTL	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-MTL	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

1026 goals are computed efficiently through iterative subtraction of current rewards, as established in our
1027 theoretical analysis.

1028 One important advantage of MOGCSL lies in its scalability to higher-dimensional objectives. By rep-
1029 resenting multiple objectives as a single vector input, MOGCSL naturally integrates additional goals
1030 without increasing architectural complexity. This vectorized formulation enhances scalability relative
1031 to conventional multi-objective approaches, which typically experience rapidly growing complexity
1032 with dimensionality. Importantly, this design eliminates the need for architectural modifications or
1033 substantial computational overhead when scaling to higher dimensions, as demonstrated in Table 2.
1034

1035 C.2 DISCUSSION ON LEARNING PARADIGM

1036 MOGCSL is designed for automated and effective learning in multi-objective settings by leveraging
1037 offline trajectories conditioned on successfully achieved multi-dimensional goals. The key intuition
1038 is to reframe the challenge of balancing potentially conflicting objectives into the more tractable task
1039 of learning to achieve realistic goals directly from observed trajectories. During training, MOGCSL
1040 directly learns to reach certain goals across multiple objectives by utilizing each trajectory in the
1041 training data as a demonstration of successful achievement of the goal that it actually achieves.
1042

1043 At inference, MOGCSL models the distribution of achievable goals across multiple objectives
1044 conditioned on the initial state. From this distribution, it samples goals that are both realistic and
1045 aligned with practical business requirements, as specified through a customizable utility function. By
1046 always grounding goal selection in attainable outcomes, MOGCSL naturally respects inter-objective
1047 trade-offs and avoids the need for manually imposed constraints or heuristic weight tuning. This
1048 design yields a more robust and flexible framework for multi-objective optimization, readily adaptable
1049 to diverse application scenarios.

1050 C.3 DISCUSSION ON GENERALIZABILITY

1051 From the experiments and discussions in Section 4.4, we observed that different goal-choosing
1052 strategies perform differently on datasets with varying reward distributions. However, we emphasize
1053 our finding that simpler, statistics-based goal-selection strategies already outperform existing baselines
1054 across diverse scenarios, demonstrating MOGCSL’s practical applicability especially where lower
1055 latency or reduced inference complexity is desired. The advanced CVAE-based algorithm provides
1056 additional performance gains specifically on datasets with higher goal values (as shown in Appendix
1057 B.8), offering practitioners the flexibility to choose strategies based on dataset characteristics and
1058 performance-complexity requirements. Rather than limiting generalizability, this observation provides
1059 valuable insights into appropriate strategy selection tailored to specific use cases in practice.
1060

1061 C.4 DISCUSSION ON CONTRIBUTION AND NOVELTY

1062 To further clarify the contribution and impact of our research, we summarize several important
1063 innovations that distinctly address critical challenges in general goal-conditional supervised learning
1064 and multi-objective recommendation scenarios:
1065

- 1066 • We are the first to integrate an MOMDP approach into GCSL’s framework, extending the goal from
1067 a scalar to a multidimensional vector. This extension is non-trivial since defining and selecting
1068 achievable "high" multi-dimensional goals is conceptually complex and previously unexplored. Our
1069 paper proposes and evaluates both principled and heuristic goal-selection approaches, demonstrating
1070 their effectiveness across different datasets.
- 1071 • We provide the first theoretical characterization of achievable goal distributions within the multi-
1072 objective setting of GCSL, formalized through Theorem 1 with complete proof (Section 3.3 and
1073 Appendix A).
- 1074 • Building upon this theoretical foundation, we developed a novel goal-choosing algorithm that
1075 models the distribution of achievable goals over interaction sequences to select desirable high
1076 goals for inference. This algorithm is novel even for the single-objective case and addresses a
1077 fundamental challenge in GCSL applications.
- 1078 • We were the first to demonstrate GCSL’s capability to mitigate the harmful effects of noisy instances,
1079 which are common in real-world recommendation data. We showed that MOGCSL can leverage

1080 the future impact of current actions across multiple objectives to achieve this ability (Section 3.4
1081 and Appendix B.2).

1083 C.5 DISCUSSION ON GENERATIVE MODELS

1084
1085 As illustrated in Section 3.3, we chose CVAE for the goal-selection algorithm primarily because
1086 of its simplicity, efficiency, and stable training dynamics, which align closely with our objective
1087 of developing scalable methods suitable for real-world recommendation scenarios. Apart from
1088 that, we’ve conducted some preliminary experiments with diffusion-based methods such as DDPO,
1089 and the results showed no significant performance improvement over CVAE. Additionally, CVAE
1090 demonstrated consistently stable training dynamics and faster inference, essential attributes for
1091 practical deployments.

1092 C.6 DISCUSSION ON DATA ROBUSTNESS

1093
1094 As illustrated in Section 3.2, MOGCSL assumes the availability of reward signals within sequential
1095 data. In recommender systems, such reward signals typically derive naturally and easily from explicit
1096 user feedback, such as clicks, purchases, or engagement durations.

1097 However, clear rewards might not be readily available in some other scenarios. For these situations,
1098 we propose the following discussions:

- 1099
1100 • Proxy rewards: Intermediate signals often effectively serve as proxies. For instance, browsing
1101 duration, add-to-cart events, or return visits in e-commerce scenarios can reliably approximate user
1102 satisfaction when explicit purchase data is sparse.
- 1103 • Synthetic rewards: In cases like path planning or robotic control, synthetic metrics such as proximity
1104 to target states or successful task completion checkpoints can be constructed to substitute explicit
1105 rewards.
- 1106 • Reward inference: Techniques such as inverse reinforcement learning or reward inference from
1107 demonstration could integrate seamlessly into MOGCSL, enabling application in less structured
1108 environments.

1109
1110 Furthermore, as discussed in Section 3.4, one of the MOGCSL’s distinct advantages lies in leveraging
1111 long-term goals to effectively handle potentially noisy immediate reward signals. Our experiments in
1112 Appendix B.2 demonstrate that this mechanism significantly enhances robustness, enabling effective
1113 learning even when datasets contain ambiguous or imprecise signals.

1114 C.7 DISCUSSION ON LIMITATIONS AND FUTURE WORKS

1115
1116 Although MOGCSL demonstrates strong effectiveness and efficiency for multi-objective recommen-
1117 dation, we identify some potential limitations and corresponding future works.

1118
1119 First, the most prominent gains from the CVAE-based goal-selection strategy are observed on datasets
1120 with relatively high average goals. Simpler statistics-based strategies remain effective when average
1121 goals are low and offer practitioners flexibility in trading off performance against computational
1122 complexity. Nonetheless, further improving the CVAE-based goal-selection mechanism is a valuable
1123 direction. Potential approaches include data augmentation techniques (e.g., re-/down-sampling,
1124 active learning) to alleviate sparsity in the high-goal region. In addition, extending MOGCSL to an
1125 online training paradigm is a promising way to further mitigate this issue if an online environment is
1126 accessible: one can pre-train on offline data, deploy the policy to collect new trajectories with higher
1127 achieved goals, and iteratively retrain on this enriched dataset.

1128
1129 Second, our current experiments focus on using long-term multi-objective outcomes to downweight
1130 noisy instantaneous behaviors that are weak indicators of positive preference. A natural extension
1131 is to directly model and leverage explicit negative signals. This could be realized by encoding
1132 negative feedback (e.g., quits, skips, dislikes) as additional dimensions in the goal vector, and then
1133 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

1134 experimentation would provide additional evidence of practical impact and constitute an important
1135 direction, especially from an industry perspective.

1136
1137 Fourth, the theoretical analysis in this paper centers on formalizing the achievable goal distribution
1138 within the multi-objective GCSL framework, which constitutes a novel conceptual and methodological
1139 advancement for MOGCSL as well as the general GCSL problem. Nonetheless, additional formal
1140 analysis, such as theoretical comparisons with other multi-objective learning methods, could further
1141 clarify the benefits and limitations of the proposed framework.

1142 Finally, while our study is motivated by empirical challenges in large-scale recommender systems
1143 (e.g., efficiency constraints and noisy interaction data), the MOGCSL formulation is broadly applicable
1144 to other multi-objective learning tasks. By reframing multi-objective optimization as learning to
1145 reach achievable multi-dimensional goals, the framework can potentially be applied to other domains,
1146 such as medical prediction, robotic control, and marketing optimization.

1147 D LLM USAGE

1148 Throughout the preparation of this paper, LLMs were used exclusively for improving clarity of
1149 expression and correcting typographical or grammatical errors. No other substantive assistance was
1150 employed.
1151
1152

1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187