SMONAC: Supervised Multiobjective Negative Actor–Critic for Sequential Recommendation

Fei Zhou, Biao Luo, *Senior Member, IEEE*, Zhengke Wu, and Tingwen Huang, *Fellow, IEEE*

***Abstract*— Recent research shows that the sole accuracy metric may lead to the homogeneous and repetitive recommendations for users and affect the long-term user engagement. Multiobjective reinforcement learning (RL) is a promising method to achieve a good balance in multiple objectives, including accuracy, diversity, and novelty. However, it has two deficiencies: neglecting the updating of negative action Q values and limited regulation from the RL Q-networks to the (self-)supervised learning recommendation network. To address these disadvantages, we develop the supervised multiobjective negative actor–critic (SMONAC) algorithm, which includes a negative action update mechanism and multiobjective actor–critic mechanism. For the negative action update mechanism, several negative actions are randomly sampled during each time updating, and then, the offline RL approach is utilized to learn their Q values. For the multiobjective actor–critic mechanism, accuracy, diversity, and novelty Q values are integrated into the scalarized Q value, which is used to criticize the supervised learning recommendation network. The comparative experiments are conducted on two real-world datasets, and the results demonstrate that the developed SMONAC achieves tremendous performance promotion, especially for the metrics of diversity and novelty.**

***Index Terms*— Actor–critic, reinforcement learning (RL), sequential recommendation system (SRS), supervised learning.**

I. Introduction

With the rapid growth of Internet in recent decades, our choice space becomes more and more diverse and varied. Meanwhile, we are often surrounded by massive options, which is the so-called paradox of choice [1]. To alleviate the severe information overloading problem, the recommendation system plays an increasingly important role while we are utilizing some virtual tools, such as social networking, e-commerce platforms, and short-video apps [2], [3], [4]. With the breakthrough of deep learning and reinforcement learning (RL), the sequential recommendation system (SRS) has made rapid development and has been utilized widely in our life. SRS usually adopts the paradigm of encoder–decoder, where the encoder jointly encodes user profile and user consumption sequences into the statement representation, and the decoder interprets it to the personalized recommendation. Because user consumption sequences are encoded continuously, the SRS learns the shifting of user preferences in time. Inspired by natural language processing technology, SRS is initially based on self-supervised learning (SL), where the learning target is to minimize the difference between the recommendation item and the next moment item in the sequence. With the rapid development of deep RL (DRL) [4], [5], [6], the SRS based on DRL has also made some progress recently because it optimizes the long-term rewards of the system [5].

Since the inadequacy of recommendation system, it tries to recommend the relevant items with the recorded sequences for users, and the accuracy has become the golden evaluation metric of system performance [7]. However, this kind of single evaluation criterion diminishes user selection space, influences user long-term engagement, and finally leads to the so-called filter bubble [8]. In the long run, if the system often recommends similar products, users will gradually feel bored, which will severely affect user experience. For instance, if you buy a jacket, then the system still recommends other kinds of jackets for you in the later, and the marginal benefit will greatly decrease while recommending a matching pair of pants or shoe is more likely to satisfy you. We find that users prefer diverse and novel products from a long-term perspective. In addition, diversity and novelty also contain intrinsic relationships, and the more diverse recommendations are likely to be more novel [9].

For SL-based SRS, it is not easy to optimize multiple evaluation metrics. We need to model it as a multiobjective optimization problem, and it usually involves the theory of differential equation, which is not easy to operate. On the contrary, the RL-based SRS is relatively easier to satisfy multiple criteria because different rewards are assigned according to different objectives. Unfortunately, RL-based SRS also faces certain practical and theoretical difficulties. RL can be divided into online RL and offline RL, and both online and offline RL-based SRS are important research directions now. The SRS based on online RL needs to build a user simulator to replace real users for training because it is impractical for users to complete massive interactions with agents during training processing [10]. User simulator imitates users' feedback to recommendations, and users can be substituted with simulator during online training. However, the accuracy of the simulator

Manuscript received 8 March 2023; revised 30 July 2023; accepted 5 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62373375 and Grant 62022094, and in part by the Zhejiang Laboratory under Grant 2021NB0AB01. *(Corresponding author: Biao Luo.)*

Fei Zhou, Biao Luo, and Zhengke Wu are with the School of Automation, Central South University, Changsha 410000, China (e-mail: feizhoucsu@csu.edu.cn; biao.luo@hotmail.com; wuzhengkecsu@gmail.com).

Tingwen Huang is with the Department of Sciences, Texas A&M University at Qatar, Doha, Qatar (e-mail: tingwen.huang@qatar.tamu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3317353.

Digital Object Identifier 10.1109/TNNLS.2023.3317353

cannot satisfy the expectation until now, and it is difficult to accurately simulate the complex user behaviors. The SRS based on offline RL directly learns from the static dataset, which avoids building user simulator and trial-and-error training way. However, the research progress in this direction is still relatively slow because the update of offline RL Q values is confronted with the inherent extrapolation-error problem, and it is difficult to accurately estimate all actions' offline RL Q values [11].

Therefore, Stamenkovic et al. [12] proposed supervised multiobjective RL (SMORL) for the SRS, which combines the SL recommendation network with RL Q-network. SMORL not only has the characteristics of SL in stability and effectiveness but also has the advantages of RL in optimizing multiobjectives and long-term rewards. SL and RL networks share the identical encoding layers, but the decoder of SL network is utilized to recommend items and RL has three kinds of Q value decoders, accuracy Q value, diversity Q value, and novelty Q value decoders. When these Q values are updated, the parameters of the public encoder will be adjusted simultaneously. If these Q values are all estimated accurately, the accuracy, diversity, and novelty evaluation metrics of the system will perform better compared with sole SL-based SRS. However, the model has two deficiencies. First, it only updates the Q values of the positive actions but neglects the calculation of negative actions' Q values, which affects the fine-tuning for the public encoder. In addition, in the SMORL model, the regulation effect of RL on the recommendation system totally depends on the fine-tuning of the public encoder, which is actually very limited and inefficient.

To overcome the aforementioned two shortcomings, we propose the supervised multiobjective negative actor–critic (SMONAC) algorithm, which includes two mechanisms: negative action update mechanism and multiobjective actor–critic mechanism. To begin with, in the negative action update mechanism, because the items in the sequences are all clicked or purchased and users are just interested in very few items at a certain time, we assume that the other actions are negative actions except for the positive actions in the sequences. Under this assumption, the negative actions are assigned with relatively small reward values, and then, the Q values of these negative actions are updated by the approach of offline RL. To reduce the complexity of calculation, we only randomly sample several negative actions to calculate their Q values during each time's updating. In addition, we further develop the multiobjective actor–critic recommendation mechanism by utilizing the scalarization result of accuracy Q value, diversity Q value, and novelty Q value. Since we only use the self-supervised outputs to make recommendations, the self-supervised network actually acts as a recommendation actor. Meanwhile, the Q values of RL outputs measure the performance of the current action from the perspective of accuracy, diversity, and novelty, so they can play the role of critics. Therefore, we multiply the integration value of three kinds of Q values into the cross-entropy loss of self-supervised recommendation network, and then, the self-supervised losses of recommendation items are updated with different importance weights.

To sum up, this article mainly includes the following three contributions.

1) We propose a negative action update mechanism, where the sampled negative actions are assigned with the same small rewards and the corresponding negative Q values are updated by the offline RL method.
2) We develop a multiobjective actor–critic mechanism, in which the SL recommendation network and RL Q-network are seen as actor and critic, respectively. The integration result of accuracy, diversity, and novelty Q values acts as importance weights for the corresponding self-supervised losses.
3) We have conducted plentiful experiments on two real-world SRS datasets, and the results demonstrate that our SMONAC algorithm not only uplifts the accuracy of recommendations to some extent but also greatly boosts their diversity and novelty.

II. Related Works

A. SL-Based SRS

Because the recurrent neural network (RNN) [13], [14] is naturally suitable for handling sequential information, the SRS based on RNN has been developed extensively. For instance, user recorded sequences were extracted to statement representation by the gated recurrent unit (GRU) network in the model of GRU4Rec [15], and then, the statement was decoded into recommendation items by the decoder. In addition, GRU4Rec++ [16] was developed to address the gradient vanishing problem of GRU4Rec by proposing the loss ranking mechanism. To enhance the feature extraction ability of GRU4Rec and GRU4Rec++ models, two RNN networks were applied to learn the user and item statements in [17], where the scorings for the recommendation items rely on both of them. Similarly, the hierarchical RNN network was used in [18], which are utilized to encode the user short-term preferences at the present session and the user long-term preferences shifting across different sessions. RNN is not good at extracting features from long sequences, and convolutional neural network (CNN) [19] is relatively adept at handling with long sequences, so there are also some studies about CNN-based SRSs. In [20], a 3-D CNN network was proposed to encode the input sequences for getting better statement representation. Furthermore, Caser [21] paid more attention to global information by the special design of union level. NItNet [22] is a generative model and consists of holed convolutional layers, which aims to learn high-level representation from both long- and short-term item dependencies. Ni et al. [23] developed comparative convolutional dynamic multiattention to extract user and item features dynamically for mining user preferences totally by multiattention-based CNN. A neural time-aware recommendation network is proposed to model stationary and dynamic user preferences by feature interaction network and CNN, respectively [24]. Although the training of RNN- and CNN-based SRS algorithms is relatively simple and effective, it is difficult to optimize the long-term benefits of recommendation systems.

B. RL-Based SRS

The RL-based SRS usually obeys Markov decision processing. For example, Rendle et al. [25] used the Markov chain to extract the pairwise items transition relationship, so as to recommend the next appropriate item for each user. In the early stage, the RL-based SRS mainly depends on traditional RL methods such as dynamic programming, Monte Carlo, and temporal difference, and the relevant typical algorithms are given as follows. Bohnenberger and Jameson [26] used decision-theoretic planning to provide a list of correlated recommendations for each user. Liebman et al. [27] proposed a music recommendation algorithm based on the approach of Monte Carlo, which recommends the songs and their playlists in the same time according to the user preferences for them. WebWatcher [28] acts as an online tour guider, which provides advice for users and learns from the usage records. Taghipour et al. [29] developed a creative RL-based recommendation system paradigm based on Q-learning. Due to the lack of neural network, the feature extraction ability of RL-based SRS is limited. With the great improvement of DRL in recent years, the research about SRS based on DRL has grown up significantly [30]. Slate Markov decision processing [31] is an early DRL-based SRS model, which utilizes deep Q-network (DQN) [32] to SRS for providing slate recommendations. To avoid massive trail-and-error interactions with users, the user simulator was constructed in [33] by the generative adversarial networks model [34]. Chen et al. [35] used generative adversarial networks to simulate the user behaviors and obtain the corresponding reward function and also proposed a cascading DQN algorithm for getting accurate Q values and the reasonable recommendation policy. However, the building of user simulator is complex and its accuracy also cannot satisfy expectation, so researchers try to construct offline RL-based SRS. The most of possible offline RL-based SRS schemes were discussed systematically in [36], including dual constraints, policy constraints, supervised regularization, support constraints, and reward extrapolation approaches. To optimize long-term gains while improving recommendation effects, in [37] and [38], the offline RL Q-network was used to adjust the encoder of SL network, and it also achieves better performance compared with sole SL or offline RL-based SRSs. Wang et al. [39], [40] proposed incremental RL and lifelong incremental RL, which reinforce the agent's adaptation ability to dynamic environments. It is a promising direction for RL-based recommendation system because users are diverse and ever-changing. The above RL-based SRSs all aim to optimize long-term gains or recommendation accuracy, neglecting the objectives of recommendation diversity and novelty, which may lead to homogeneous recommendation problems.

C. Multiobjective Recommendation

Improving the diversity and novelty of recommendation system is supposed as important research direction for recommendation system. Early works on this direction concentrated on postprocessing approaches, which attempt to balance the accuracy and diversity [41], [42], [43]. To alleviate the problems of great cumulative loss on the ranking function, personalized ranking ways were developed in [44]. Chen et al. [45] attempted to address the problems of the pairwise measurement way on diversity and the neglecting of the intrinsic relationship between items, by developing the model of determinantal point process [46] that learns the correlation among items utilizing a kernel matrix. When the learning of the matrix was finished, a diverse item list can be generated from it by sampling items [45], [47], [48]. These models all fulfill the objective of balancing the accuracy and diversity at most. Hirata et al. [49] tried to solve the k-MIPS problem by the IP-Greedy algorithm, which controls the diversity of recommendations by a parameter. Vrijenhoek et al. [50] treated the diversity as a divergence score, which observes the difference between the recommendations and the specific target. In the RL setting, Zheng et al. [51] aimed to improve diversity, by randomly sampling items around the current recommendation. To generate an appropriate and diverse recommendation list, diversify top-N recommendation with fast Monte Carlo tree search [52] generates recommendations with the way of Monte Carlo tree search and provides plans with actor–critic networks. Hansen et al. [53] developed a ranker, which provided a list of ranked and diverse items. This model is actually a simple ranker, and it itself does not learn to generate a diverse list of items, while its learning is based on the way of REINFORCE [54]. Finally, some prior recommendation system research about optimizing multiobjectives depended on the theory of Pareto optimization, which utilized grid search [55] or multigradient descent [56]. However, one Pareto optimal solution cannot exceed other Pareto solutions all-roundly with the consideration of all objectives.

III. Methodology

We first expound the modeling methods of SL recommendation network and RL Q-network in this section. Subsequently, the negative action mechanism and multiobjective actor–critic mechanism are developed subsequently. Finally, the training processing of SMONAC is presented in Algorithm 1. The whole structure of SMONAC and the overview of the proposed two mechanisms are shown in Fig. 1.

A. Modeling of SL and RL Networks

In the SMONAC algorithm, we only utilize the outputs of SL network to generate recommendations, and the Q value outputs of RL networks are used for the regulation of SL recommendation network. Note that the RL networks have three output ends, which correspond to the accuracy, diversity, and novelty Q values. Therefore, to get the recommendation items and three kinds of Q values, SL and RL networks should be modeled in the SMONAC algorithm.

1) Modeling of SL Network: The SL-based SRS is to recommend the desiring items for users at the next moment based on the prior recorded sequences by utilizing a self-supervised network. To effectively learn the representation from the previous user sequences, the encoding and decoding networks are detached and they are used to extract state representation and generate recommendations. The learning of

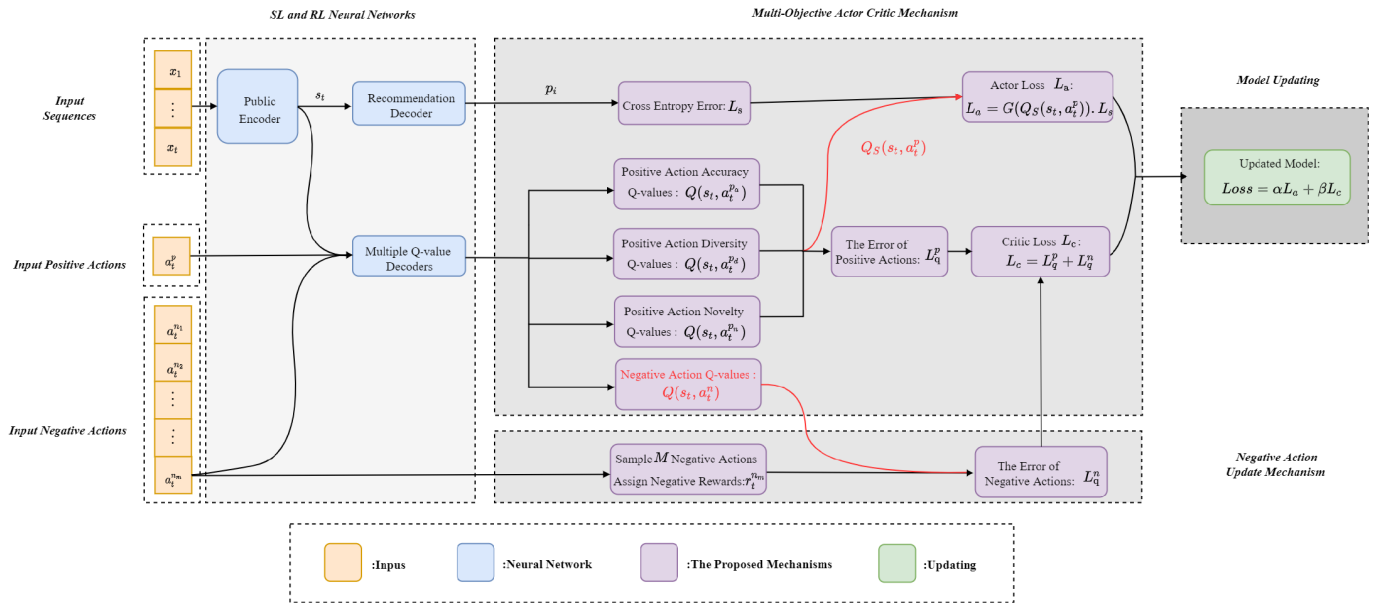

Fig. 1. Illustration diagram of SMONAC. The past sequences and actions are input into the public encoder, and the corresponding predictions and three kinds of Q values are generated by the SL network and RL networks, respectively. Two proposed mechanisms are shown in pink grids. Negative actions' Q values are updated by the negative action update mechanism. Accuracy, diversity, and novelty Q values are first integrated by multiobjective actor–critic mechanism, and it is subsequently multiplied into the loss of SL network.

SL-based SRS depends on self-supervised labels, namely, next moment clicked or purchased items in the sequences, where its objective is to minimize the loss between the predicted recommendations and self-supervised signals. Note that the encoder–decoder is actually a general network paradigm, not a concrete network structure. Factually, almost all SL-based SRSs are utilizing encoder–decoder network paradigm. For instance, RNN-based GRU4Rec, CNN-based Caser, and NItNet all employ the form of encoder–decoder.

As aforementioned, the SL recommendation network employs the paradigm of encoder–decoder, in which the encoder extracts the representations from the past clicked and purchased items $x_{1:t}$, and the decoder translates the representations into the next moment recommendations. Therefore, the statement $s_t = E(x_{1:t})$, where $E(\cdot)$ means the encoder network. Subsequently, the decoder transforms the state representation into probabilities of all possible recommendations as follows:

$$p_j = \frac{e^{o_j}}{\sum_{k=1}^{n} e^{o_k}} \tag{1}$$

where $o_j = D(s_t)$, D represents the decoder network, and n equals the output dimension. The cross-entropy loss of SL recommendation network is calculated as follows:

$$L_s = \sum_{i=1}^{N} \sum_{i=1}^{n} y_j \log\left(p_j\right) \tag{2}$$

where N represents the value of training batch size and $y_j = 1$ if and only if j equals the index of clicked or purchased item; otherwise, $y_j = 0$. L_s is updated by the stochastic gradient descent (SGD) method.

2) Modeling of RL Networks: As shown in Fig. 1, RL Q-networks also utilize the form of encoder–decoder. The public encoder is designed for encoding the past user recorded sequences into the current statement, while the decoder is used to compute the Q values at the present statement and action. We use the self-supervised network to provide the recommendation items, while Q-network is utilized to compute the Q values. Meanwhile, the SL and RL networks share the same public encoder. When the Q value is updated by the way of gradient backpropagation, the parameters of the public encoder will also be adjusted simultaneously so that the self-supervised recommendation network also has the advantages of RL in optimizing the long-term rewards of system.

Since we optimize the accuracy, diversity, and novelty of recommendations at the same time, we need to set accuracy reward r_t^{pa}, diversity reward r_t^{pd}, and novelty reward r_t^{pn}. Note that we just utilize the offline dataset to directly update the corresponding three kinds of Q values $Q(s_t, a_t^{\text{pa}})$, $Q(s_t, a_t^{\text{pd}})$, and $Q(s_t, a_t^{\text{pn}})$, so the Q values are learned by the approach of offline RL. Offline RL still models the SRS as a Markov sequential decision processing. Therefore, the statement S, action A, reward R, state transition function P, and discount factor γ are defined as follows.

1) *Statement S:* The statement $s_t = E(x_{1:t})$, which is the encoding output of the past sequences. The SL recommendation decoder and offline RL Q value decoders share the identical statement s_t.
2) *Action A:* Items in the sequences are clicked or purchased by users, so we define the items in the sequences as positive actions, which means that x_{t+1} is the positive action a_t^p. In SMONAC, we presume that the other possible actions except for a_t^p at certain states are all seen as negative actions a_t^n.
3) *Reward R:* We set the reward of positive action as r_t^p and the reward of negative action as r_t^n. In fact, r_t^p is a vector, which contains the elements r_t^{pa}, r_t^{pd}, and

r_t^{pn}. Specifically, r_t^{pa}, r_t^{pd}, and r_t^{pn} are positive rewards from the aspects of accuracy, diversity, and novelty, respectively.

4) *Transition Function P:* $p(s_{t+1}|s_t, a_t)$ represents the occurrence probability of next statement s_{t+1}, while the agent adopts the action a_t at the statement s_t. In $p(s_{t+1}|s_t, a_t)$, $s_{t+1} = E(x_{1:t} + a_t)$.
5) *Discount Factor* γ: $\gamma \in [0, 1]$ represents the value of agents importance to long-term gain, where $\gamma = 0$ implies that the agent totally concentrates on the current gain, and $\gamma = 1$ is the highest importance value of the long-term gain.

In fact, when we are introducing the reward set $\{r_t^{\text{pa}}, r_t^{\text{pd}}, r_t^{\text{pn}}, r_t^n\}$, we do not give the specific definition of them. r_t^{pa} is the accuracy reward for the click or purchase item in the sequence. If a_t^{pa} is the click item, $r_t^{\text{pa}} = r_{\text{click}}$; otherwise $r_t^{\text{pa}} = r_{\text{purchase}}$. r_{click} and r_{purchase} are the constant reward values for the click and purchase items from the perspective of accuracy. r_t^{pd} is the diversity reward of the top prediction item, which is defined as follows:

$$r_t^{\text{pd}} = 1 - \cos\left(e^{l_t}, e^{p_t}\right) \tag{3}$$

where $\cos(\cdot)$ represents the operation of cosine, $e^{(\cdot)}$ is the corresponding item's embedding, l_t and p_t are the last item in the session and top prediction item in recommendations, respectively, and r_t^{pn} is the novelty reward for the click or purchase item in the sequence. We categorize the items that are frequently interacted with users into popular item set, and the rest items are classified into unpopular item set. If a_t^p is the in popular item set, $r_t^{\text{pn}} = 0$; otherwise, $r_t^{\text{pn}} = 1$. r_t^n is the reward for negative action, and we set it as 0 in this article.

RL agent seeks the best behavior policy $\pi(a|s)$, which makes the agent choose the best action $a \in A$ at statement $s \in S$, so as to maximize the long-term cumulative rewards

$$\max_{\pi(a|s)} E_{\tau\sim\pi}(R(\tau)) \tag{4}$$

where $R(\tau) = \sum_{t=0}^{|\tau|} \gamma r_t^p$ and $\tau = (s_0, a_0, s_1, a_1, \ldots)$ represents the action trajectory of the agent, which are got by implementing actions according to the behavior policy $\pi(a|s)$.

B. Negative Action Update Mechanism

Since the past recorded sequence has already been encoded into the latent statement s_t by public network $E(\cdot)$, s_t is subsequently input into RL output layers for the calculation of Q values. The design of public encoder facilitates the transfer of knowledge and the fine-tuning from the multiple RL ends to the SL recommendation end. Therefore, the Q values are formulated by output layers as follows:

$$Q(s_t, a_t) = \delta\left(s_t h_t^T + b\right) \tag{5}$$

where δ means the activation function and h_t and b are transformation matrices and biases, respectively, which are also trainable parameters of the Q value output layers.

For the Q values of positive actions $Q(s_t, a_t^p)$, they are learned subsequently because they are orderly in the sequences. Note that $Q(s_t, a_t^p)$ is a vector, which is composed of $Q(s_t, a_t^{\text{pa}})$, $Q(s_t, a_t^{\text{pd}})$, and $Q(s_t, a_t^{\text{pn}})$. We utilize DQN to learn the Q values, which is updated by the Bellman equation. In fact, the updating of DQN is based on value-based methods, where the target action is chosen by $a'_p = \arg\max Q_S(s_{t+1}, a_{t+1}^p)$. The DQN adopts the one-step temporal difference updating method to learn the Q value, and the target Q value is calculated as follows:

$$T_p\left(s_t, a_t^p\right) = r_t^p + \max_{a'_p} Q\left(s_{t+1}, a'_p\right) \tag{6}$$

where r_t^p and $Q(s_{t+1}, a'_p)$ are both vectors. r_t^p contains the accuracy, diversity, and novelty rewards r_t^{pa}, r_t^{pd}, and r_t^{pn}, and $Q(s_t, a'_p)$ includes the three corresponding Q values $Q(s_{t+1}, a_{t+1}^{\text{pa}})$, $Q(s_{t+1}, a_{t+1}^{\text{pd}})$, and $Q(s_{t+1}, a_{t+1}^{\text{pn}})$.

The preprocessed sequences only contain clicked or purchased items. If we only update the positive action Q values $Q(s_t, a_t^p)$ in the sequences, the Q values of negative actions cannot be updated, so as to influence the regulation effect form the RL Q-network to the SL network. Since users are just interested in a small portion of items at certain statement, we presume that all items except a_t^p at the present statement are negative actions a_t^n. Subsequently, the negative actions are assigned with appropriate small rewards $r_t^{n_m}$. We still utilize DQN and temporal difference approaches to update the Q values of negative actions $Q(s_t, a_t^n)$. The corresponding update target $T_n(s_t, a_t^{n_m})$ is computed as follows:

$$T_n\left(s_t, a_t^{n_m}\right) = r_t^{n_m} + \max_{a'_{n_m}} Q\left(s_t, a'_{n_m}\right) \tag{7}$$

where $a'_{n_m} = \arg\max Q(s_t, a_t^{n_m})$. Note that we set $T_n(s_t, a_t^{n_m}) = r_t^{n_m} + \max_{a'_{n_m}} Q(s_t, a'_{n_m})$ instead of $T_n(s_t, a_t^{n_m}) = r_t^{n_m} + \max_{a'_{n_m}} Q(s_{t+1}, a'_{n_m})$. This is because we do not know whether users will click or purchase the negative actions, so the statement at the next moment s_{t+1} is actually uncertain. Therefore, $Q(s_{t+1}, a'_{n_m})$ is substituted by $Q(s_t, a'_{n_m})$, so as to obtain an approximate target value. To reduce the computational complexity, we only sample M negative actions $a_t^{n_m}$ at once to update their Q values, where m is the corresponding index.

Because we have obtained the target values of positive and negative actions $T_p(s_t, a_t^p)$ and $T_n(s_t, a_t^{n_m})$, the mean square error of positive and negative actions' Q values L_q is computed as follows:

$$\begin{aligned} L_q &= L_q^p + L_q^n \\ &= \sum_{i=1}^{N} \Bigg[\left(T_p\left(s_t, a_t^p\right) - Q\left(s_t, a_t^p\right)\right)^2 \\ &\qquad + \sum_{m=1}^{M} \left(T_n\left(s_t, a_t^{n_m}\right) - Q\left(s_t, a_t^{n_m}\right)\right)^2 \Bigg] \end{aligned} \tag{8}$$

where $(T_p(s_t, a_t^p) - Q(s_t, a_t^p))^2$ is the loss of the positive action Q values, namely L_q^p, and $\sum_{m=1}^{M} (T_n(s_t, a_t^{n_m}) - Q(s_t, a_t^{n_m}))^2$ is the loss of M negative action Q values, namely L_q^n. Therefore, we utilize the SGD method to minimize L_q. In this way, not only $Q(s_t, a_t^p)$ is updated, but also $Q(s_t, a_t^{n_m})$ is learned so that the more accurate adjustment from the RL output ends to the public encoder is achieved. We call this kind of update effect from the negative Q values as a negative action update mechanism.

C. Multiobjective Actor–Critic Mechanism

Although we calculated the Q values of negative actions $Q(s_t, a_t^{n_m})$ by the negative action update mechanism, the regulation effect from the RL Q-networks to the SL recommendation network is still limited. It is noteworthy that whether we update the Q values of positive action or negative action, the regulation effect is only reflected on the fine-tuning of public encoder parameters. Therefore, to exploit the regulation effect from the RL Q-networks to SL recommendation sufficiently, we consider further utilizing the Q values to regulate the SL network. In fact, we only use the SL network to recommend items, so it acts as a recommendation actor. Because the Q values in the RL are used to evaluate the current recommendation and actions, our RL networks here can be regarded as critics. Therefore, when the actor network recommends items, we utilize the Q values of the critic networks to criticize them, so as to assign different importance weights for different recommendations during their updating processing.

However, our critic network actually has three outputs, and they criticize the quality of the current positive action from the perspective of accuracy, diversity, and novelty. To integrate the evaluation effect of the three Q values and simplify the calculation processing, we choose to integrate $Q(s_t, a_t^{\text{pa}})$, $Q(s_t, a_t^{\text{pd}})$, and $Q(s_t, a_t^{\text{pn}})$ to scalarized Q value $Q_S(s_t, a_t^p)$, so as to criticize the current positive action in the later. $Q_S(s_t, a_t^p)$ is obtained by multiplying with scalarization function $f = \omega$ as follows:

$$Q_S\left(s_t, a_t^p\right) = f\left(Q\left(s_t, a_t^p\right)\right) = \omega^T Q\left(s_t, a_t^p\right) \tag{9}$$

where ω is the parameter vector of the scalarization function f.

Because $Q_S(s_t, a_t^p)$ is the integration result of $Q(s_t, a_t^{\text{pa}})$, $Q(s_t, a_t^{\text{pd}})$, and $Q(s_t, a_t^{\text{pn}})$, it can comprehensively reveal the accuracy, diversity, and novelty performance of the current recommendations. Consequently, we multiply the gradient stopped $Q_S(s_t, a_t^p)$ to the SL network loss L_s for getting the regulated loss L_a, namely actor loss, as follows:

$$L_a = G\left(Q_S\left(s_t, a_t^p\right)\right) \cdot L_s \tag{10}$$

where $G(\cdot)$ means freezing the updating of trainable parameters in $Q_S(s_t, a_t^p)$ to avoid the disturbance for the updating of SL and RL networks. Because the purchased items usually have higher Q values than the clicked items, the minimizing of L_a better differentiates the clicked and purchased items compared with optimizing L_s, and then, the system concentrates more on the purchased item during the training processing. We call this kind of regulation from the RL network to the SL network as a multiobjective actor–critic mechanism.

D. Training Processing of SMONAC

Algorithm 1 demonstrates the specific training processing of SMONAC, which mainly is divided into four stages. To begin with, we need to implement the training preparing. Subsequently, we calculate the loss of SL and RL networks. After certain steps' pretraining for getting stable Q values, we conduct the main training stage.

Algorithm 1 SMONAC Algorithm

Input: Self-supervised network S, Q-network Q, Target Q-network Q', Replay buffer B, Item set χ, Pretraining max-steps T

Output: Trainable parameters

1: Initialize all trainable parameters
2: **while** *not converged* **do**
3: Sample the tuples $(x_{1:t}, a_t^p, r_t^p, x_{1:t+1})$ from the B
4: Compute r_t^{pa}, r_t^{pd} and r_t^{pn} respectively
5: Encode the sequences $x_{1:t}$ and $x_{1:t+1}$ into s_t and s_{t+1}
6: Sample M negative actions $a_t^{n_m}$ from the χ
7: Allocate $r_t^{n_m}$ for $a_t^{n_m}$
8: Compute the self-supervised loss:
$$L_s = \sum_{i=1}^{N} \sum_{j=1}^{n} y_j \log(p_j)$$
9: Calculate the positive action's target:
$$T_p(s_t, a_t^p) = r_t^p + \max_{a'_p} Q(s_{t+1}, a'_p)$$
10: Calculate the negative action's target:
$$T_n(s_t, a_t^{n_m}) = r_t^{n_m} + \max_{a'_{n_m}} Q(s_t, a'_{n_m})$$
11: Compute the critic's loss:
$$L_q = \sum_{i=1}^{N} [(T_p(s_t, a_t^p) - Q(s_t, a_t^p))^2 + \sum_{m=1}^{M} (T_n(s_t, a_t^n) - Q(s_t, a_t^{n_m}))^2]$$
12: **# Stage1: pretraining stage for stable Q-values**
13: Compute the pretraining loss: $L_{pre} = L_s + \beta L_q$
14: Update the model by the gradient: ∇L_{pre}
15: **# Stage2: training with two proposed mechanisms**
16: Calculate the integration Q value:
$$Q_S(s_t, a_t^p) = f(Q(s_t, a_t^p)) = \omega^T Q(s_t, a_t^p)$$
17: Compute the actor loss: $L_a = G(Q_S(s_{t+1}, a_{t+1}^p)).L_s$
18: Compute the training loss: $L_{SMONAC} = \alpha L_a + \beta L_q$
19: Update the model by the gradient: ∇L_{SMONAC}
20: **end while**

Steps 1–6 of Algorithm 1 correspond to the training preparing stage, which includes initialization, data sampling, and statement encoding. To ensure randomness of training, the neural networks need to conduct parameter initialization. Before training, the source data have been sorted as the type of four-element tuples $(x_{1:t}, a_t^p, r_t^p, x_{1:t+1})$, and we randomly sample a batch of tuples for each time training in step 3. Subsequently, in steps 4 and 5, we compute the corresponding accuracy, diversity, and novelty rewards for positive actions, and, simultaneously, input $x_{1:t}$ and $x_{1:t+1}$ into the public encoder for obtaining s_t and s_{t+1}. In addition, we sample the negative actions from the item set χ and allocate the negative rewards $r_t^{n_m}$ for them in steps 6 and 7.

Before the pretraining or main training stage of SMONAC, the losses of SL and RL networks are calculated from steps 8 to 11. We first calculate the loss of SL recommendation network L_s, namely actor loss. Steps 9 and 10 compute the positive and negative actions' update targets $T_p(s_t, a_t^p)$ and $T_n(s_t, a_t^{n_m})$, respectively. Step 11 calculates the corresponding loss of Q values L_q, namely critic loss. Note that r_t^p is

actually a vector, which contains the corresponding elements of accuracy, diversity, and novelty.

In the multiobjective actor–critic mechanism, we first need to calculate the scalarized positive Q value $Q_S(s_t, a_t^p)$ and multiply it to L_s. However, at the beginning of learning, the estimation of positive Q values is inaccurate. If $Q_S(s_t, a_t^p)$ is unstable, it will cause the disturbance to even cause a divergence in the system. To avoid this situation, we do not compute $Q_S(s_t, a_t^p)$ during the pretraining stage and utilize it to criticize L_s. The pretraining stage varies from steps 12 to 14. As we have obtained the losses of SL recommendation network and RL Q-network, the total loss L_{pre} in the pretraining stage is calculated as follows:

$$L_{\text{pre}} = L_s + \beta L_q \tag{11}$$

where β is the controlling coefficient for the update of RL Q-networks. Therefore, the model is updated by the gradient ∇L_{pre} in step 14. We observe that only the negative action update mechanism is working in the pretraining stage, and a multiobjective actor–critic mechanism does not play a role here.

In the main training stage, the Q values become relatively steady, and the two proposed mechanisms both work now. Specifically, the calculation of negative action Q values makes the fine-tuning for the parameters of public encoder more precisely. In addition, the criticizing from the integration Q value $Q_S(s_t, a_t^p)$ to L_s makes the agent assign different importance values to them according to their Q values. Steps 15–19 reveal the main operations in this stage. $Q_S(s_t, a_t^p)$ is calculated by multiplying with the scalarization function $f = \omega$ in step 16. Subsequently, the actor loss L_a is computed by multiplying with gradient stopped $Q_S(s_t, a_t^p)$ according to the multiobjective actor–critic mechanism. Therefore, the total loss L_{SMONAC} in the main training stage is formulated in step 18 as follows:

$$L_{\text{SMONAC}} = \alpha L_a + \beta L_q \tag{12}$$

where α is the coefficient for controlling the criticizing from the scalarization Q value to the SL network. The whole model is updated by the gradient ∇L_{SMONAC} in the later, and it will continue its updating until the converge of the model. Furthermore, we observe that the SMONAC algorithm can directly integrate with the majority of SL-based SRS with the paradigm of encoder–decoder or merely change slightly, so it is widely applicable in the SRS.

IV. Experiments

We conduct extensive experiments on two real-world recommendation system datasets to evaluate the proposed SMONAC algorithm for answering four research questions as follows.

1) *RQ1:* How does the SMONAC perform comparing with other state-of-the-art (SOTA) SRS algorithms?
2) *RQ2:* How do the negative action update mechanism and multiobjective actor–critic mechanism influence the recommendation effect separately?
3) *RQ3:* How does the value of negative actions M and negative reward r_t^n influence the effect of negative action update mechanism?
4) *RQ4:* How do the controlling coefficients α and β affect the updating of SL recommendation network and RL Q-network, respectively?

TABLE I
Detailed Information About R2 and RC15

Dataset	Sequences	Items	Clicks	Purchases
R2	195,523	70,852	1,176,680	57,629
RC15	200,000	26,702	1,110,965	43,946

A. Experimental Settings

1) Datasets: The experiments are conducted on the following two sequential recommendation datasets R2 and RC15. The more detailed information of these two datasets is shown in Table I.

a) R2: It is sorted out from a real-world recommendation system dataset: RetailRocket,[1] which is the user consuming records on an e-commerce website. The operations of viewing and adding-to-cart are regarded as clicking and purchasing on R2, respectively, and the operation of transaction is eliminated from the RetailRocket. The items that are interacted smaller than three times are removed and the sessions whose length is smaller than 3 are also discarded.

b) RC15: This dataset is derived from the RecSys Challenge 2015,[2] which includes click and purchase items in the sequences. Identically, the sessions whose length is smaller than 3 are discarded, and the preprocessed RC15 contains 200k sequences.

The clicked or purchased items on two datasets are grouped by Userid and sorted by Timestamp, respectively, and then, we obtain the consecutive sequences, which only include clicked or purchased items. To facilitate the learning of the SMONAC, we further sort the sequences into the form of four-element tuples $(x_{1:t}, a_t^p, r_t^p, x_{1:t+1})$. The dataset is split by the approach of cross validation. The proportions of training data, evaluation data, and test data are 80%, 10%, and 10%, respectively. In the stage of evaluation and test, the recommendation item is compared with the ground-truth item in the sequence one by one, and the whole item set is used for ranking. All experiments are implemented five times with five different random seeds, and the presented results are average values.

2) Baselines: We combine the SMONAC with three SOTA sequential recommendation models as follows.

1) *GRU4Rec [15]:* The GRU4Rec uses the GRU network to extract the latent statement representation and then translates it into the recommendation item by a decoder network.
2) *Caser [21]:* Caser utilizes the CNN network to extract the statement from the embedding-matrix of the sequences. This algorithm first brings forward the concept of union level, which facilitates the learning of global features.
3) *NItNet [22]:* The residual design is introduced to this model for improving the learning ability of neural

[1]RetailRocket: https://www.kaggle.com/retailrocket/ecommerce-dataset
[2]RecSys Challange 2015: https://recsys.acm.org/recsys15/challenge/

TABLE II

TOP-k PERFORMANCE OF SMONAC INTEGRATED WITH DIFFERENT BACKBONES COMPARED WITH DIFFERENT SOTAS ON R2. NG MEANS NDCG, AND K IS TAKEN FROM THE SET {10, 20}. THE BEST RESULTS HAVE BEEN SHOWN IN BOLD

Models	Accuracy				Diversity				Novelty				Repetitiveness		
	HR-10	NG-10	HR-20	NG-20	CV-1	CV-5	CV-10	CV-20	CV-1	CV-5	CV-10	CV-20	R-5	R-10	R-20
GRU	0.2673	0.1878	0.3082	0.1981	0.2439	0.4695	0.5699	0.6632	0.1837	0.4139	0.5238	0.6267	14.25	29.44	60.59
GRU-SQN	0.2967	0.2094	0.3406	0.2205	0.2180	0.4114	0.4975	0.5763	0.1526	0.3489	0.4430	0.5299	14.62	30.19	62.22
GRU-SMORL	0.3060	0.2103	0.3535	0.2224	0.2796	0.5369	0.6419	0.7353	0.2154	0.4871	0.6029	0.7064	13.53	28.02	57.89
GRU-SMONAC	**0.3339**	**0.2350**	**0.3836**	**0.2475**	**0.3169**	**0.5707**	**0.6724**	**0.7625**	**0.2518**	**0.5241**	**0.6367**	**0.7366**	**11.63**	**24.13**	**49.56**
Caser	0.2302	0.1675	0.2628	0.1758	0.2327	0.4379	0.5133	0.5718	0.1643	0.3773	0.4605	0.5252	16.16	33.24	68.39
Caser-SQN	0.2454	0.1778	0.2803	0.1867	0.2088	0.3880	0.4511	0.5021	0.1387	0.3219	0.3914	0.4479	16.88	34.50	70.58
Caser-SMORL	0.2657	0.1898	0.3052	0.1998	0.2855	0.5411	0.6324	0.7138	0.2224	0.4917	0.5925	0.6827	15.90	32.47	66.76
Caser-SMONAC	**0.2770**	**0.1983**	**0.3181**	**0.2089**	**0.3219**	**0.5757**	**0.6722**	**0.7576**	**0.2617**	**0.5300**	**0.6366**	**0.7311**	**14.75**	**30.51**	**63.07**
NItNet	0.3007	0.2060	0.3506	0.2186	0.2867	0.5113	0.6033	0.6837	0.2305	0.4595	0.5605	0.6495	12.25	25.76	54.00
NItNet-SQN	0.3129	0.2150	0.3586	0.2266	0.2802	0.5255	0.6077	0.6750	0.2184	0.4747	0.5651	0.6395	12.27	25.93	54.47
NItNet-SMORL	0.3183	0.2222	0.3659	0.2342	0.3429	0.6335	0.7351	0.8129	0.2800	0.5938	0.7062	0.7924	10.92	22.89	47.73
NItNet-SMONAC	**0.3373**	**0.2375**	**0.3867**	**0.2498**	**0.3513**	**0.6417**	**0.7420**	**0.8211**	**0.2850**	**0.6018**	**0.7131**	**0.8013**	**10.47**	**21.77**	**45.23**

TABLE III

TOP-k PERFORMANCE OF SMONAC INTEGRATED WITH DIFFERENT BACKBONES COMPARED WITH DIFFERENT SOTAS ON RC15. NG MEANS NDCG, AND K IS TAKEN FROM THE SET {10, 20}. THE BEST RESULTS HAVE BEEN SHOWN IN BOLD

Models	Accuracy				Diversity				Novelty				Repetitiveness		
	HR-10	NG-10	HR-20	NG-20	CV-1	CV-5	CV-10	CV-20	CV-1	CV-5	CV-10	CV-20	R-5	R-10	R-20
GRU	0.3793	0.2279	0.4581	0.2478	0.2481	0.4330	0.5188	0.5942	0.1777	0.3707	0.4654	0.5492	12.11	25.63	53.24
GRU-SQN	0.3946	0.2394	0.4741	0.2587	0.2406	0.4025	0.4710	0.5364	0.1656	0.3363	0.4122	0.4849	12.20	25.81	53.47
GRU-SMORL	0.4007	0.2433	0.4793	0.2632	0.2825	0.4758	0.5577	0.6334	0.2086	0.4176	0.5086	0.5927	11.29	23.81	48.88
GRU-SMONAC	**0.4168**	**0.2531**	**0.4962**	**0.2737**	**0.6008**	**0.8310**	**0.8774**	**0.9052**	**0.5592**	**0.8076**	**0.8638**	**0.8947**	**2.52**	**6.28**	**16.26**
Caser	0.3593	0.2177	0.4371	0.2372	0.2631	0.4349	0.5019	0.5608	0.1912	0.3724	0.4466	0.5120	14.38	29.65	60.73
Caser-SQN	0.3668	0.2223	0.4448	0.2420	0.2154	0.3525	0.4057	0.4557	0.1411	0.2810	0.2154	0.3953	14.45	29.79	60.82
Caser-SMORL	0.3664	0.2224	0.4425	0.2417	0.3174	0.5157	0.5944	0.6685	0.2476	0.4621	0.5495	0.6316	13.77	28.56	58.52
Caser-SMONAC	**0.3757**	**0.2287**	**0.4532**	**0.2483**	**0.3548**	**0.5725**	**0.6645**	**0.7563**	**0.2884**	**0.5252**	**0.6273**	**0.7293**	**13.57**	**28.25**	**57.83**
NItNet	0.3885	0.2332	0.4684	0.2535	0.2950	0.4914	0.5705	0.6427	0.2313	0.4354	0.5228	0.6030	10.03	22.02	46.84
NItNet-SQN	0.4083	0.2492	0.4878	0.2693	0.2737	0.4572	0.5183	0.5715	0.2082	0.3975	0.4649	0.5239	10.19	22.32	47.26
NItNet-SMORL	0.4116	0.2505	0.4898	0.2703	0.3385	0.5639	0.6518	0.7283	0.2720	0.5156	0.6131	0.6981	9.97	21.73	45.49
NItNet-SMONAC	**0.4209**	**0.2576**	**0.4991**	**0.2775**	**0.3510**	**0.5709**	**0.6637**	**0.7419**	**0.2834**	**0.5293**	**0.6233**	**0.7216**	**9.50**	**20.43**	**44.19**

network, and the dilated CNN is used here for the getting larger receptive domain.

These three algorithms all can be combined with SMONAC. To better demonstrate the performance of our algorithm, we also make comparisons with SQN [37] and SMORL [12], which are SOTA single- and multiple-objective SRS algorithms, respectively. Note that SQN and SMORL are both integration algorithm and can be combined with other SOTA SL-based SRS models, which contains GRU4Rec, Caser, and NItNet in this article.

3) Metrics: The accuracy, diversity, and novelty are three optimizing objectives for our SRS from different perspectives, so we should design different metrics for them.

a) Accuracy metrics: We choose the HR@K (Hit Ratio) and NDCG@K (normalized discounted cumulative gain [57]) as the metrics for measuring the accuracy. HR@K(click) implies whether the clicked items are among the top-k recommendations. We define it as follows:

$$\mathrm{HR@}K(\mathrm{click}) = \frac{1}{N_T^c}\sum_{i=1}^{N_T^c} h\left(\mathrm{top}_k\right) \tag{13}$$

where N_T^c equals the amount of all sessions with click action a_t^p in the test dataset and top_k are the top k predicted items. $h(\cdot)$ equals 1, while top_k contains the click action a_t^p; otherwise, it equals 0. HR@K(purchase) has a similar definition, and we just need to substitute the purchase for the click. The first and the kth item in the recommendations have the same effect in the metric HR@K(click), but NDCG@K(click) is a weighted metric, which is relatively sensitive metric for different positions. The predicted item will be assigned higher scoring when it is ranked in a higher position of top-k recommendations. NDCG@K(purchase) has a similar definition, and we just need to substitute the purchase for the click. To evaluate the stability and robustness of SMONAC, we set K as 10 and 20 subsequently.

b) Diversity metrics: When we are introducing diversity, it actually has two kinds of meanings: individual diversity and aggregate diversity. The individual diversity indicates the diversity for each time recommendation. For example, if the system recommends k dissimilar items in one recommendation, we can say that the recommendation is very diverse at this time. However, if the system recommends the same or similar items in each time's recommendation, the total diversity is actually very low. Therefore, we utilize the aggregate diversity to measure the diversity in this article, and the aggregate diversity metric CV@K is defined as follows:

$$\mathrm{CV@}K = \frac{1}{C_T}\sum_{i=1}^{N_T} c\left(\mathrm{top}_k\right) \tag{14}$$

where C_T is the classes of item set, $c(\cdot)$ equals the new item classes of top-k predictions, and CV@K indicates the coverage rate of click item classes in recommendations compared with total click item classes in the item set.

c) Repetitiveness metrics: In fact, the direct novelty metric has a similar definition with diversity metric, where the denominator is substituted with the total click item classes in the less popular items. We also define an indirect metric of novelty, which is repetitiveness metric and is utilized to measure the degree of filter bubble. It is defined as follows:

$$R@K = \frac{1}{N_T}\sum_{i=1}^{N_T^c} \mathrm{rep}\left(\mathrm{top}_k\right) \tag{15}$$

where N_T is the amount of all sessions in the test dataset and rep$(\cdot)$ equals the number of repetitive items in top-k predictions. The higher value of repetitiveness metric means the lower novelty of recommendations.

4) Parameter Settings: For two datasets, the input sequences are consisted of the recent ten items before the target item. If the length of sequence is smaller than 10, we complement the rest with the same padding item. All networks are optimized by the Adam optimizer [58], and the size of the batch is set as 256. The learning rates are set as 0.01 and 0.005 for RC15 and R2, respectively. The item embedding size is set as 64 for all networks to guarantee a fair comparison. The hidden statement size is set as 64 in the model of GRU4Rec. For Caser, we utilize one vertical convolution filter and 16 horizontal filters, whose heights are sampled from the set of $\{2, 3, 4\}$. The dropout ratio is set as 0.1. For NextItNet, we keep the same parameters as the original paper. The discount factor γ is set as 0.5, which is recommended in [37].

B. Performance Comparison

Tables II and III show the experimental results of SMONAC and its comparative SOTA models on R2 and RC15, respectively. The best results have been shown in bold. There are several observations that we obtain from Tables II and III.

1) The HR@K(click) and NDCG@K(purchase) results of SMONAC improve stably compared with the results of sole SL-based SRS models GRU4Rec, Caser and NItNet, and SL combined with RL SRS models SQN and SMORL on both two datasets. The results show that the calculation of the negative actions Q values better fine-tunes the parameters of public encoder, and the critical effect from RL Q values to the SL recommendations helps the learning of SL network.
2) In most situations, the SMONAC dramatically promotes the diversity and novelty performance in comparison with the corresponding results of GRU4Rec, Caser, NItNet, SQN, and SMORL models on R2 and RC15. The remarkable boost indicates that the proposed negative action update mechanism and the multiobjective actor–critic mechanism play an important role in improving the ability of network to generate diverse and novel recommendations.
3) The $R@K$(click) values of SMONAC decrease with different ranges compared with the results of GRU4Rec, Caser, NItNet, SQN, and SMORL models on both two datasets. The performance of SMONAC in the metric of repetitiveness illustrates its ability in scaling down the filter bubble and promotes the number of effective recommendations.
4) From another perspective, the proposed SMONAC algorithm improves the performance of all SOTAs in the accuracy, diversity, novelty, and repetitiveness metrics with different K's on R2 and RC15. The all-round boost implies the stability and robustness of SMONAC. Furthermore, SMONAC can be integrated with most SL-based SRS algorithms with the promotion of recommendation effect, which indicates that the SMONAC is very applicable and practical.

TABLE IV
ABLATION EXPERIMENTS OF THE PROPOSED TWO MECHANISMS ON R2

Models	HR@20	CV-Div@20	CV-Nov@20	R@20
GRU	0.3082	0.6632	0.6267	60.59
GRU-SMORL	0.3535	0.7353	0.7064	57.89
GRU-SMONRL	0.3667	0.7435	0.7197	53.59
GRU-SMOAC	0.3701	0.7521	0.7238	51.67
GRU-SMONAC	**0.3836**	**0.7625**	**0.7366**	**49.56**

TABLE V
ABLATION EXPERIMENTS OF THE PROPOSED TWO MECHANISMS ON RC15

Models	HR@20	CV-Div@20	CV-Nov@20	R@20
GRU	0.4581	0.5942	0.5492	53.24
GRU-SMORL	0.4793	0.6334	0.5927	48.88
GRU-SMONRL	0.4870	0.6713	0.6395	41.26
GRU-SMOAC	0.4904	0.8926	0.7993	23.25
GRU-SMONAC	**0.4962**	**0.9052**	**0.8947**	**16.26**

C. Ablation Experiments

To illustrate the individual effects of the negative action update mechanism and multiobjective actor–critic mechanism, we choose GRU4Rec as the basic SOTA to implement the ablation experiments on R2 and RC15. We compare the effects of GRU4Rec, GRU4Rec with SMORL algorithm (GRU-SMORL), GRU4Rec with negative action update mechanism (GRU-SMONRL), GRU4Rec with multiobjective actor–critic mechanism (GRU-SMOAC), and GRU4Rec with SMONAC algorithm (GRU-SMONAC). The HR@20, CV-Div@20, CV-Nov@20, and R@20 results of ablation experiments on R2 and RC15 are shown in Tables IV and V, and we observe the phenomena from Tables IV and V as follows.

1) For accuracy, diversity, and novelty metrics, GRU-SMONRL and GRU-SMOAC both perform better than GRU-SMORL model, which indicates that the negative action update mechanism and the multiobjective actor–critic mechanism both promote the multiobjective recommendation effect.
2) From Tables IV and V, we also see that the multiobjective actor–critic mechanism plays a more important role than negative action update mechanism in promoting multiobjective recommendation metrics.
3) For all multiobjective metrics on R2 and RC15 datasets, GRU-SMONAC performs better than GRU-SMONRL and GRU-SMOAC, which means that the effects of the proposed mechanisms are superimposed in the SMONAC algorithm.

D. Numbers and Rewards of Negative Actions

The number of sampled negative actions M and the values of negative rewards r_t^n are both influence factors for the performance of negative action update mechanism. Therefore, we test different numbers of sampled negative actions and values of negative rewards for finding the best ones and also evaluating the robustness of SMONAC simultaneously. It is noteworthy that all experiments take the GRU4Rec as the backbone in this section.

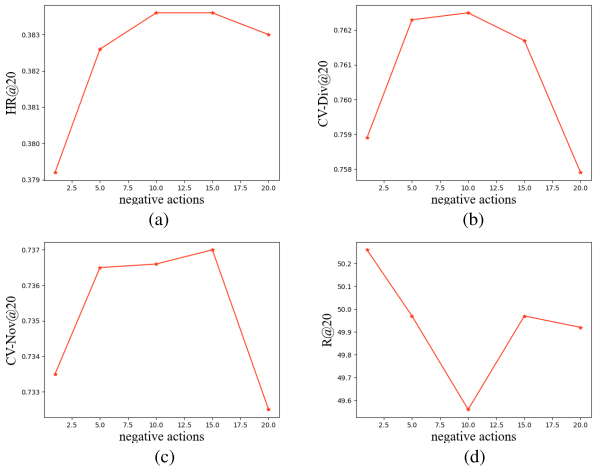

Fig. 2. Accuracy, diversity, novelty, and repetitiveness results of GRU-SMONAC algorithm with a different number of negative actions M on R2. (a) R2-HR@20. (b) R2-CV-Div@20. (c) R2-CV-Nov@20. (d) R2-R@20.

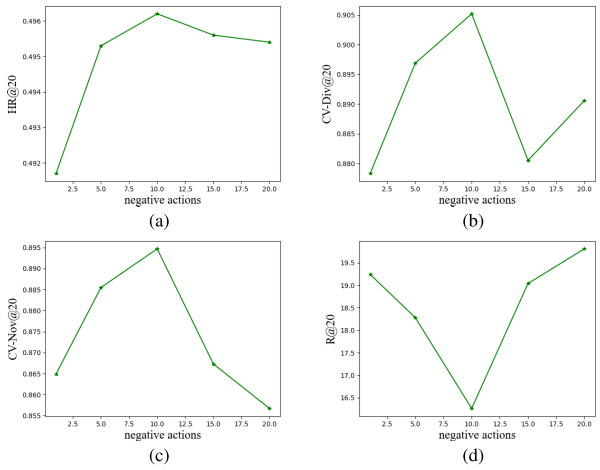

Fig. 3. Accuracy, diversity, novelty, and repetitiveness results of GRU-SMONAC algorithm with a different number of negative actions M on RC15. (a) RC15-HR@20. (b) RC15-CV-Div@20. (c) RC15-CV-Nov@20. (d) RC15-R@20.

1) Negative Actions M: To reduce the computational complexity and improve the training speed, we sample M negative actions for each time updating in the negative action update mechanism. We take a value from the set $\{1.0, 5.0, 10.0, 15.0, 20.0\}$, and the corresponding results about the metrics of accuracy, diversity, novelty, and repetitiveness on two datasets are shown in Figs. 2 and 3. The following observations can be seen from them.

1) The results show that the SMONAC is not very sensitive to the number of negative actions, and it implies that the negative action update mechanism is robust and stable for the number change of sampled negative actions.
2) Although different numbers of negative actions M do not influence the performance a lot, the number $M = 10$ performs best on all the above metrics on R2 and RC15. Therefore, we set M as 10 for all other experiments while sampling the negative actions.

2) Negative Rewards r_t^n: r_t^n is the reward for the negative action. To evaluate the performance of the SMONAC for

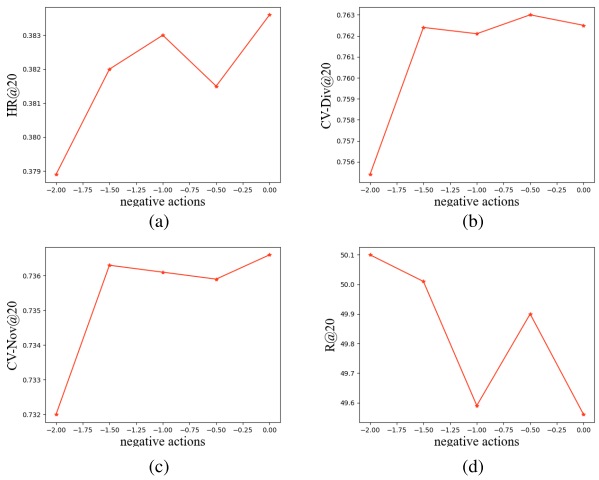

Fig. 4. Accuracy, diversity, novelty, and repetitiveness results of GRU-SMONAC algorithm with different settings of negative rewards r_t^n on R2. (a) R2-HR@20. (b) R2-CV-Div@20. (c) R2-CV-Nov@20. (d) R2-R@20.

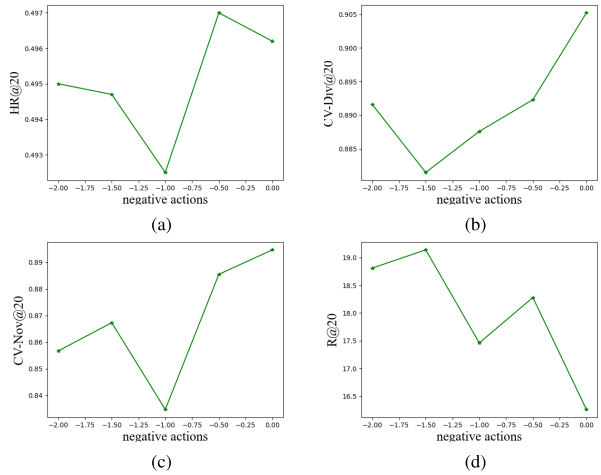

Fig. 5. Accuracy, diversity, novelty, and repetitiveness results of GRU-SMONAC algorithm with different settings of negative rewards r_t^n on RC15. (a) RC15-HR@20. (b) RC15-CV-Div@20. (c) RC15-CV-Nov@20. (d) RC15-R@20.

different settings of negative rewards r_t^n, we test five parameters of r_t^n in the set of $\{0.0, -0.5, -1.0, -1.5, -2.0\}$, and the results about accuracy, diversity, novelty, and repetitiveness on R2 and RC15 are shown in Figs. 4 and 5. From Figs. 4 and 5, we can get the following observations.

1) To begin with, the change of negative reward does not affect the SMONAC a lot. We suppose that the main factor in this phenomenon is probably that relatively small rewards for negative actions are essential, but the specific values do not have great importance.
2) At the same time, the minor fluctuation of the results for different negative rewards implies that the negative reward update mechanism is robust for the change of negative reward.
3) We also find that the setting of $r_t^n = 0$ performs better than others. Consequently, we choose $r_t^n = 0$ as negative reward in other experiments.

TABLE VI
SETTINGS OF COEFFICIENT α ON R2

α	HR@20	CV-Div@20	CV-Nov@20	R@20
0.5	**0.3836**	**0.7625**	**0.7366**	49.62
1.0	0.3825	0.7616	0.7310	**49.56**
1.5	0.3818	0.7601	0.7298	49.67
2.0	0.3816	0.7612	0.7289	49.89

TABLE VII
SETTINGS OF COEFFICIENT β ON R2

β	HR@20	CV-Div@20	CV-Nov@20	R@20
0.5	0.3823	0.7609	0.7313	49.67
1.0	**0.3836**	**0.7625**	**0.7366**	**49.56**
1.5	0.3814	0.7621	0.7498	50.11
2.0	0.3817	0.7604	0.7386	50.45

TABLE VIII
SETTINGS OF COEFFICIENT α ON RC15

α	HR@20	CV-Div@20	CV-Nov@20	R@20
0.5	**0.4962**	**0.9052**	**0.8947**	**16.26**
1.0	0.4923	0.8989	0.8817	17.26
1.5	0.4918	0.8971	0.8923	19.63
2.0	0.4825	0.8843	0.8864	18.89

TABLE IX
SETTINGS OF COEFFICIENT β ON RC15

β	HR@20	CV-Div@20	CV-Nov@20	R@20
0.5	**0.4969**	0.8923	0.8898	17.78
1.0	0.4962	**0.9052**	**0.8947**	16.26
1.5	0.4948	0.9012	0.8933	**16.23**
2.0	0.4925	0.9008	0.8894	16.89

E. Coefficients of α and β

The coefficient α is utilized to control the critical effect from the Q values to the recommendation network, the coefficient β is used to control the updating of positive and negative action Q values, and we set α and β as $\{0.5, 1.0, 1.5, 2.0\}$ subsequently for obtaining the best values. At the same time, we hope to evaluate the robustness of SMONAC. It is noteworthy that all experiments take the GRU4Rec as the backbone here. The results of different α and β values about accuracy, diversity, novelty, and repetitiveness on R2 and RC15 are shown in Tables VI–IX.

1) The results show that the SMONAC is relatively stable to the change of α and β, and it indicates that the SMONAC is very robust for the variance of α and β.
2) Although different values of α and β just affect the performance of SMONAC slightly, the coefficients $\alpha = 0.5$ and $\beta = 1.0$ perform best on all aforementioned metrics on R2 and RC15. Therefore, we set α and β as 0.5 and 1.0, respectively, for all other experiments.

V. CONCLUSION

In this article, we develop the SMONAC algorithm, which mainly contains two mechanisms, namely the negative action update mechanism and the multiobjective actor–critic mechanism. The negative action update mechanism is designed to update the Q values of negative actions by the approach of offline RL, so as to better fine-tune the parameters of public encoder. The multiobjective actor–critic mechanism utilizes the integration Q value of accuracy, diversity, and novelty Q values to criticize the SL recommendation network, which aims to help the SL recommendation network differentiate the actions and recommendations with different importance. We implemented the comparison experiments on two real-world SRS datasets, and the results demonstrate that the SMONAC improves the recommendation accuracy and dramatically promotes the diversity and novelty of recommendations compared with SOTA baselines such as SQN and SMORL models. In addition, the ablation experiments show that the proposed two mechanisms can promote multiobjective recommendation effects, and the effects of two mechanisms are superimposed in the SMONAC algorithm. Furthermore, the SMONAC shows its robustness for the change of SL-based SRS baselines, and it performs steadily for the variance of negative action numbers, negative reward values, and controlling coefficients.

REFERENCES

[1] B. Schwartz, "The paradox of choice: Why more is less," *Positive Psychol. Pract.*, vol. 17, no. 32, pp. 143–150, 2012.
[2] J. Han et al., "Adaptive deep modeling of users and items using side information for recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 737–748, Mar. 2020.
[3] W.-D. Xi, L. Huang, C.-D. Wang, Y.-Y. Zheng, and J.-H. Lai, "Deep rating and review neural network for item recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6726–6736, Nov. 2022.
[4] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Jan. 2020.
[5] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proc. 12th ACM Conf. Recommender Syst.*, Sep. 2018, pp. 95–103.
[6] T. Hu, B. Luo, C. Yang, and T. Huang, "MO-MIX: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12098–12112, Oct. 2023.
[7] D. Wang, X. Zhang, D. Yu, G. Xu, and S. Deng, "CAME: Content- and context-aware music embedding for recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1375–1388, Mar. 2021.
[8] E. Pariser, *The Filter Bubble: What the Internet is Hiding From You*. London, U.K.: Penguin, 2011.
[9] D. M. Fleder and K. Hosanagar, "Recommender systems and their impact on sales diversity," in *Proc. 8th ACM Conf. Electron. Commerce*, Jun. 2007, pp. 192–199.
[10] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, "Reinforcement learning to optimize long-term user engagement in recommender systems," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2810–2818.
[11] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2052–2062.
[12] D. Stamenkovic, A. Karatzoglou, I. Arapakis, X. Xin, and K. Katevas, "Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, Feb. 2022, pp. 957–965.
[13] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[15] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, *arXiv:1511.06939*.
[16] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 843–852.
[17] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," 2019, *arXiv:2001.04830*.

[18] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 130–137.

[19] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[20] T. X. Tuan and T. M. Phuong, "3D convolutional networks for session-based recommendation with content features," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 138–146.

[21] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 565–573.

[22] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 582–590.

[23] J. Ni, Z. Huang, C. Yu, D. Lv, and C. Wang, "Comparative convolutional dynamic multi-attention recommendation model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3510–3521, Aug. 2022.

[24] Q. Zhang, L. Cao, C. Shi, and Z. Niu, "Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5125–5137, Oct. 2022.

[25] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 811–820.

[26] T. Bohnenberger and A. Jameson, "When policies are better than plans: Decision-theoretic planning of recommendation sequences," in *Proc. 6th Int. Conf. Intell. User Interfaces (IUI)*, Jan. 2001, pp. 21–24.

[27] E. Liebman, M. Saar-Tsechansky, and P. Stone, "DJ-MC: A reinforcement-learning agent for music playlist recommendation," 2014, *arXiv:1401.1880*.

[28] T. Joachims, D. Freitag, and T. Mitchell, "Webwatcher: A tour guide for the world wide web," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 1997, pp. 770–777.

[29] N. Taghipour, A. Kardan, and S. S. Ghidary, "Usage-based web recommendations: A reinforcement learning approach," in *Proc. ACM Conf. Recommender Syst.*, Oct. 2007, pp. 113–120.

[30] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–38, Jul. 2023.

[31] P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin, and B. Coppin, "Deep reinforcement learning with attention for slate Markov decision processes with high-dimensional states and actions," 2015, *arXiv:1512.01124*.

[32] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] J. Gauci et al., "Horizon: Facebook's open source applied reinforcement learning platform," 2018, *arXiv:1811.00260*.

[34] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[35] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 1052–1061.

[36] T. Xiao and D. Wang, "A general offline reinforcement learning framework for interactive recommendation," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2021, vol. 35, no. 5, pp. 4512–4520.

[37] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Self-supervised reinforcement learning for recommender systems," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 931–940.

[38] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Supervised advantage actor-critic for recommender systems," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, Feb. 2022, pp. 1186–1196.

[39] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, "Incremental reinforcement learning with prioritized sweeping for dynamic environments," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 621–632, Apr. 2019.

[40] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[41] A. Ashkan, B. Kveton, S. Berkovsky, and Z. Wen, "Optimal greedy diversity for recommendation," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 15, 2015, pp. 1742–1748.

[42] L. Qin and X. Zhu, "Promoting diversity in recommendation by entropy regularizer," in *Proc. 23rd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2013, pp. 2698–2704.

[43] C. Sha, X. Wu, and J. Niu, "A framework for recommending relevant and diverse items," in *Proc. 23rd Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 16, 2016, pp. 3868–3874.

[44] P. Cheng, S. Wang, J. Ma, J. Sun, and H. Xiong, "Learning to recommend accurate and diverse items," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 183–192.

[45] L. Chen, G. Zhang, and E. Zhou, "Fast greedy map inference for determinantal point process to improve recommendation diversity," in *Proc. 32nd Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 5627–5638.

[46] F. Lavancier, J. Møller, and E. Rubak, "Determinantal point process models and statistical inference," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 77, no. 4, pp. 853–877, Sep. 2015.

[47] R. Warlop, J. Mary, and M. Gartrell, "Tensorized determinantal point processes for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1605–1615.

[48] M. Wilhelm, A. Ramanathan, A. Bonomo, S. Jain, E. H. Chi, and J. Gillenwater, "Practical diversified recommendations on Youtube with determinantal point processes," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 2165–2173.

[49] K. Hirata, D. Amagata, S. Fujita, and T. Hara, "Solving diversity-aware maximum inner product search efficiently and effectively," in *Proc. 16th ACM Conf. Recommender Syst.*, Sep. 2022, pp. 198–207.

[50] S. Vrijenhoek, G. Bénédict, M. G. Granada, D. Odijk, and M. De Rijke, "RADio—Rank-aware divergence metrics to measure normative diversity in news recommendations," in *Proc. 16th ACM Conf. Recommender Syst.*, Sep. 2022, pp. 208–219.

[51] G. Zheng et al., "DRN: A deep reinforcement learning framework for news recommendation," in *Proc. World Wide Web Conf. World Wide Web*, 2018, pp. 167–176.

[52] L. Zou, L. Xia, Z. Ding, D. Yin, J. Song, and W. Liu, "Reinforcement learning to diversify top-N recommendation," in *Proc. 24th Int. Conf. DASFAA*, 2019, pp. 104–120.

[53] C. Hansen, R. Mehrotra, C. Hansen, B. Brost, L. Maystre, and M. Lalmas, "Shifting consumption towards diverse content on music streaming platforms," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 238–246.

[54] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.

[55] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, "Pareto-efficient hybridization for multi-objective recommender systems," in *Proc. 6th ACM Conf. Recommender Syst.*, Sep. 2012, pp. 19–26.

[56] N. Milojkovic, D. Antognini, G. Bergamin, B. Faltings, and C. Musat, "Multi-gradient descent for multi-objective recommender systems," 2019, *arXiv:2001.00846*.

[57] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

Fei Zhou received the B.E. degree from the Department of Mechanical Engineering and Automation, Shantou University, Shantou, China, in 2018. He is currently pursuing the master's degree with the Department of Automation, Central South University, Changsha, China.

His research interests include reinforcement learning and recommender systems.

Biao Luo (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Beihang University, Beijing, China, in 2014.

He is currently a Professor with the School of Automation, Central South University (CSU), Changsha, China. Before joining CSU, he was an Associate Professor and an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, from 2014 to 2018. His current research interests include intelligent control, reinforcement learning, deep learning, and decision-making.

Dr. Luo is the Vice-Chair of the Adaptive Dynamic Programming and Reinforcement Learning Technical Committee, Chinese Association of Automation. He serves as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Artificial Intelligence Review*, *Neurocomputing*, and the *Journal of Industrial and Management Optimization*.

Zhengke Wu received the B.E. degree in measuring and control technology and instrumentations from Henan University, Kaifeng, China, in 2020. He is currently pursuing the M.E. degree in artificial intelligence with the School of Automation, Central South University (CSU), Changsha, China.

His current research interests include human in the loop, imitation learning, and deep reinforcement learning.

Tingwen Huang (Fellow, IEEE) received the B.S. degree from Southwest Normal University (now Southwest University), Chongqing, China, 1990, the M.S. degree from Sichuan University, Chengdu, China, 1993, and the Ph.D. degree from Texas A&M University, College Station, TX, USA, in 2002.

After graduation, he worked as a Visiting Assistant Professor with Texas A&M University. In August 2003, he joined Texas A&M University at Qatar (TAMUQ) as an Assistant Professor, where he was promoted to a Professor in 2013. He is currently a Professor at Texas A&M University at Qatar, Doha, Qatar. His research interests include neural networks, chaotic dynamical systems, complex networks, and optimization and control.