Handling Varied Objectives by Online Decision Making

Lanjihong Ma
National Key Laboratory for Novel Software Technology, School of Artificial Intelligence, Nanjing University
maljh@lamda.nju.edu.cn

Zhen-Yu Zhang
Center for Advanced Intelligence Project, RIKEN
zhen-yu.zhang@riken.jp

Yao-Xiang Ding
State Key Lab of CAD & CG, Zhejiang University
dingyx.gm@gmail.com

Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology, School of Artificial Intelligence, Nanjing University
zhouzh@lamda.nju.edu.cn

Abstract

Conventional machine learning typically assume a fixed learning objective throughout the learning process. However, for real-world tasks in open and dynamic environments, objectives can change frequently. For example, in autonomous driving, a car has several default modes, but a user's concern for speed and fuel consumption varies depending on road conditions and personal needs. We formulate this problem as *learning with varied objectives* (LVO), where the goal is to optimize a dynamic weighted combination of multiple sub-objectives by sequentially selecting actions that incur different losses on these sub-objectives. We propose the VaRons algorithm, which estimates the action-wise performance on each sub-objective and adaptively selects decisions according to the dynamic requirements on different sub-objectives. Further, we extend our approach to cases involving contextual representations and propose the ConVaRons algorithm, assuming parameterized linear structure that links contextual features to the main objective. Both the VaRons and ConVaRons are provably minimax optimal with respect to the time horizon T, with ConVaRons showing better dependency with the number of sub-objectives K. Experiments on dynamic classifier and real-world cluster service allocation tasks validate the effectiveness of our methods and support our theoretical findings.

CCS Concepts

• **Computing methodologies** → **Online learning settings**.

Keywords

dynamic objective, online decision making, open-environment machine learning, learning with varied objectives

ACM Reference Format:
Lanjihong Ma, Zhen-Yu Zhang, Yao-Xiang Ding, and Zhi-Hua Zhou. 2024. Handling Varied Objectives by Online Decision Making. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3637528.3671812

KDD '24, August 25–29, 2024, Barcelona, Spain.

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671812

1 Introduction

Conventional machine learning workflows typically have a predefined and fixed learning objective throughout the learning process, such as maximizing accuracy. However, the learning objectives in real-world applications could *vary continuously* due to changes in the specific task at hand. Considering model updating in autonomous driving, the main objective could be a *dynamic combination* of several sub-objectives, such as increasing speed and improving energy efficiency according to specific vehicle states, such as fuel level, road, and weather conditions. Meanwhile, classifiers trained for a fixed objective may be less effective when the objective change happens. It is important to formalize this problem and design corresponding machine learning algorithms with the ability to adapt to the arbitrarily changing learning objectives in the open-environment machine learning [45].

In this paper, we investigate the problem of learning with varying objectives, where the main objective can be represented by a dynamic combination of several sub-objectives. Consider the illustration in Figure 1: for the autonomous driving system, one can switch to "high-performance mode" when the main objective is driving performance and to "power saving mode" when the main objective changes to endurance. In practice, however, the user usually needs both performance and endurance; the main objective is neither but a weighted combination of the two. Meanwhile, the weights could vary over time according to the user's needs. In this case, the best choice may not be either modes but other customized modes. Nevertheless, it is also hardly impossible to define a default mode for each combination. Therefore, it is desirable not only to be able to learn about other custom modes but also to be able to make decisions about which modes to select according to the knowledge learned and specific requirements that may be constantly changing.

Some works consider similar problems by optimizing multiple sub-objectives simultaneously, such as *multi-objective multi-armed bandits* (MO-MAB) [11, 26] and *multi-objective optimization* (MOO) [13, 23]. MOMAB assumes the main objective is a static, predefined weighted combination of sub-objectives, which does not consider a dynamic setting where the weights can vary over time. Thus, MO-MAB approaches are unsuitable for handling varied objectives with time-varying weights.

On the other hand, the goal of MOO approaches is to identify the Pareto front, which is the set of modes where no other modes can outperform across all sub-objectives. However, as we usually have a specific main objective at each time, the best choice in most

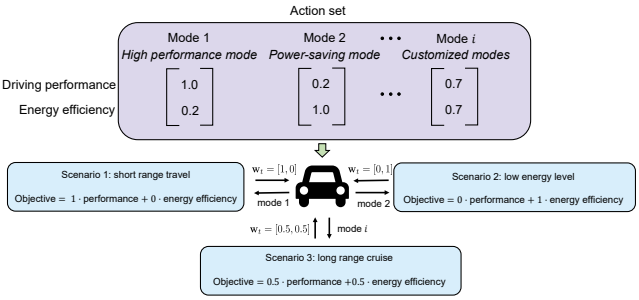


Figure 1: An illustrative example. The driver can choose "high-performance mode" when the main objective is driving performance; when the main objective varies to endurance, "power saving mode" is then a better choice; when both objectives are required, i.e., main objective $= 0.5 \cdot$ performance $+ 0.5 \cdot$ efficiency, other customized modes are better choices.

cases is unique. In this case, as the Pareto front contains all Pareto optimal modes rather than a single best choice, decisions made according to the Pareto front may not always achieve the best performance [35, 43]. We illustrate this by the example in Figure 1, where we focus on the third scenario where the two sub-objectives are equally weighted. In this case, modes 1 and 2, though both being Pareto optimal, are not the best choice compared to the customized mode i. In this case, the Pareto front found by MOO approaches may be sub-optimal due to the inability to select a single best choice.

Considering these limitations in dealing with a specific yet arbitrarily varied objective, we try to take a step towards bridging this gap by trying to answer the following question:

Can we develop a decision-making strategy that consistently guarantees the selection of a single, optimal choice, one that is adaptive to an arbitrarily varied objective?

In this paper, we formalize the problem discussed above as *learning with varied objectives* (LVO) within an online decision-making framework and provide a positive answer to the above question. Specifically, we follow the basic assumption in the works of Drugan and Nowé [11], Murata et al. [26] that the main objective can be represented by the weighted combination of several sub-objectives. Still, we additionally assume that the weights can be *continuously changing*, representing changing emphasis on different sub-objectives. We formulate this problem by the following decision-making process: the learner first defines several modes that can possibly handle sub-objectives, and we refer to these modes as the action set of the decision-making procedure if no further confusion arises. At each time, after receiving the main objective defined by the explicitly known weights, the learner chooses *a single action* from the action set. By choosing the action, feedback indicating whether this choice is good or not is returned, enabling the learner to update for better future selections. The goal is to maximize the performance (or, equivalently, minimize the loss) on the main objective by such decision-making procedure. A brief comparison between LVO, MOMAB, and MOO is given in Table 1.

Based on our formulation of the LVO problem, we propose the *varied-objective online Newton step* (VaRons) algorithm for LVO.

Table 1: Comparisons between LVO, MO-MAB and MOO.

Problem settings	LVO (Ours)	MO-MAB [11, 26]	MOO [13, 23]
Handle varied-objectives	✓	×	✓
Select single best action	✓	✓	×

In particular, since the main objective (the weights of each sub-objective) could change over time, the loss distribution on the main objective can be time-varying even for the same action, making it difficult to estimate the main objective accurately. Alternatively, we avoid directly estimating the main objective and turn to estimating the action-wise performance of each sub-objective, based on which we then utilize the weighted combination structure to recover the estimation of the main objective. According to this estimation, we then construct an estimated loss distribution over all actions, from which we employ a randomized sampling strategy to select an action. After such a choice, feedback representing loss on the main objective is returned, allowing us to update the above estimations. By this repeated estimation-sampling-updating procedure, we prove the algorithm is *minimax optimal* with respect to the total time horizon T up to logarithmic factors. Furthermore, we extend our learning problem to the contextual scenario, where actions can be featured by additional contextual information, such as the real-time states and parameters of the action. By introducing the additional parameterized linear structure connecting the contextual features and the main objectives, we propose a variant of the VarRons algorithm named *contextual varied-objective online Newton step* (ConVaRons) algorithm to handle this problem. The proposed variant takes the same randomized sampling strategy but with advanced estimation method to achieve better dependency on the number of actions K. We empirically test the proposed algorithms on dynamic classifier selection tasks, where the performance measure is changing so the corresponding optimal classifier changes over time, which is an suitable task for LVO. We further test our proposed algorithms on real-world dynamic cluster service allocation tasks, verifying the effectiveness of our proposed approaches.

2 Problem Formulation

In this section, we formalize the LVO problem, introduce our performance measure, and discuss the main challenges in this problem.

The LVO problem. We assume the main objective is a dynamic weighted combination of d pre-defined sub-objectives. Different from traditional scalarized models in multi-objective optimization where weights are fixed, our focus now is a time-varying weight $\mathbf{w}_t \in \Delta_{d-1}$[1], that represents the different focuses on the sub-objectives at each time. At each time $t \in [T] := \{1, ..., T\}$, once the weights decided by specific requirements have been obtained, the learner chooses an action a_t from a candidate action set $\mathcal{A}$, from this decision the i-th sub-objective will suffer an action-dependent loss $l_i(a_t)$. Notice the specific losses on each sub-objective are not always available; we focus on the worst case when the learner only observes the (possibly noisy) feedback y_t on the main objective $\ell_t(a_t)$, which we assume to be bounded in $[0, 1]$, such that

$$\mathbb{E}[y_t] = \ell_t(a_t) = \mathbf{w}_t^\top \mathbf{L}(a_t), \tag{1}$$

[1]We denote by Δ_{d-1} as the $(d-1)$-simplex, defined as the set of all d-dimensional vectors $[\mathbf{w}_t^1, \mathbf{w}_t^2, \ldots, \mathbf{w}_t^d]$ such that $\mathbf{w}_t^i \geq 0$ for all i, and $\sum_{i=1}^d \mathbf{w}_t^i = 1$

The general assumptions. Given an action set $\mathcal{A}$ of size K, there exists a ground truth objective vector $\mathbf{L}(a) \in [0,1]^d$ for each action $a \in \mathcal{A}$, implying the abilities of handling sub-objectives for each action.

The online decision making procedure. At each time $t = 1, \cdots, T$:

1: The learner receives the weight vector $\mathbf{w}_t \in \Delta_{d-1}$ from specific task or environment. and the corresponding main objective $\ell_t(a) = \mathbf{w}_t^\top \mathbf{L}(a)$ for all $a \in \mathcal{A}$, where $\mathbf{L}(a)$ is the loss vector of action a and is unknown to the learner.

2: The learner chooses an action $a_t \in \mathcal{A}$ and incurs loss on the main objective $\ell_t(a_t)$, which is not observed by the learner.

3: A bounded noisy feedback y_t satisfying $\mathbb{E}[y_t] = \mathbf{w}_t^\top \mathbf{L}(a_t), y_t \in [0,1]$ is observed by the learner.

The target of the learner: minimizing the cumulative (pseudo) dynamic regret over time horizon T: $\bar{\mathcal{R}}_T := \sum_{t=1}^T \mathbb{E}[\mathbf{w}_t^\top (\mathbf{L}(a_t) - \mathbf{L}(a_t^*)]$, where a_t^* is the optimal selection in hindsight defined as $a_t^* := \arg\min_{a \in \mathcal{A}} \mathbf{w}_t^\top \mathbf{L}(a)$

Figure 2: The formalized Learning with Varied Objective (LVO) by online decision making.

where $\mathbf{L}(a_t) = [l_1(a_t), \ldots, l_d(a_t)]$ is the ground truth vectorized loss on sub-objectives incurred by selecting action a_t. We formulate the detailed learning process in Figure 2.

Performance measures. We compare learner's performance against *the best decision in hindsight each time*. Specifically, we first define the instantaneous regret at time t as

$$r_t := \ell_t(a_t) - \ell_t(a_t^*) = \mathbf{w}_t^\top \left(\mathbf{L}(a_t) - \mathbf{L}(a_t^*)\right), \tag{2}$$

where a_t^* is the best action with respect to the main objective at each time t, formally defined as

$$a_t^* := \arg\min_{a \in \mathcal{A}} \ell_t(a) = \arg\min_{a \in \mathcal{A}} \mathbf{w}_t^\top \mathbf{L}(a). \tag{3}$$

Similar to most online learning works, by choosing the right action with respect to the time-varying weight vector, the learner's goal is to minimize the following cumulative (pseudo) *dynamic regret*,

$$\bar{\mathcal{R}}_T := \sum_{t=1}^T \bar{r}_t = \sum_{t=1}^T \mathbb{E}\left[\mathbf{w}_t^\top \left(\mathbf{L}(a_t) - \mathbf{L}(a_t^*)\right)\right], \tag{4}$$

which is the expected sum of the instantaneous regret over the total time horizon T, and the expectation is taken with respect to the randomness of the algorithmic realization.

Challenges. We analyze the main challenges of the LVO problem into the following three aspects:

- **Non-i.i.d. Loss Distribution**: the main objective is determined by the weight vector $\mathbf{w}_t$ at each time, which we do not put any assumption on, as it is decided by realistic requirements and tasks. The weight vector $\mathbf{w}_t$ can be arbitrarily assigned or even fully adversarial in the worst case. Consequently, *selecting the same action at different time steps may incur different losses on the main objective*, and thus, the feedback y_t does not follow a consistent distribution for the same action. This lack of consistency makes it difficult to estimate the main objective accurately.
- **Partial Observability**: the learner receives feedback that is related to the main objective of the selected action only. This implies the learner not only has to *handle the exploration-exploitation trade-off* in traditional bandit learning but also has to *handle the indirect feedback* that is only related to the main objective, without directly revealing the actual losses on the sub-objectives.
- **Dynamic Best Action**: the best action defined in (3), is dynamic with respect to the time-varying weight vector $\mathbf{w}_t$. This dynamic nature presents two main challenges. Algorithmically, the algorithm should adapt robustly to the weight at each time, as different weight corresponds to different optimal actions without a static convergence target. Technically, the regret metrics introduced in (2) and (4) are special cases of dynamic regret [47]. The analysis is much more challenging compared with the static regret that is most analyzed in existing literature. .

Independently solving one of the above challenges is relatively easy, while it is not the case when they arise simultaneously in a single problem, especially when we additionally seek the optimal approaches with provable guarantees.

3 Our approaches

In this section, we introduce the proposed *varied-objective online Newton step* (VaRons) algorithm. Theoretical analysis shows that the proposed algorithm is minimax optimal up to logarithmic factors of the time horizon T. We then extend the proposed algorithm to the contextual scenario where actions can be represented by some contextual features, and propose a variant of the VaRons algorithm and also discuss the corresponding theoretical guarantees.

3.1 VaRons for the LVO problem

Recall that according to the LVO problem formulation, at each time step, the learner receives the weights for the sub-objectives that represent the real-time requirements. After receiving the weights, the learner has to choose an action from the action set. By choosing such an action, a loss on the main objective is incurred and returned to the learner, after which the learner can learn about the chosen action and prepare for future decisions.

Specifically, the algorithm starts with the learning rate γ as input; how to choose such a learning rate will be specified later in the theorems. Throughout the whole process, the algorithm maintains two critical estimations: the estimated objective vector $\hat{\mathbf{L}}$ in (1) for each action, representing the performance of the action on all sub-objectives; and the estimated Hessian $\mathbf{H}$ that is used to approximate second-order partial derivatives of the loss function. The algorithm starts by initializing the above estimation-related variables and the regularization parameter ϵ. For each time $t \in [T] := \{1, ..., T\}$, the algorithm first receives the weight vector $\mathbf{w}_t$, which represents the specific emphasis on each sub-objective from the environment, according to which the algorithm can estimate the main objective for each action according to the current estimations. The algorithm then computes a K-dimensional distribution p_t, which

Algorithm 1 VaRons: varied-objective online Newton step

Require: learning rate $\gamma > 0$
1: **initialize:** $\epsilon = \frac{256}{d}$, $\hat{\mathbf{L}}(a) = \mathbf{0}^d$, $\mathbf{H}(a) = \epsilon I_d$ for all $a \in \mathcal{A}$
▷ variables for estimations and Hessians of each action
2: **for** $t = 1, \cdots, T$ **do**
3: Receive weight vector $\mathbf{w}_t$;
4: Estimate main objective $\hat{\ell}_t(a) = \mathbf{w}_t^\top \hat{\mathbf{L}}(a), \forall a \in \mathcal{A}$;
5: Compute $p_t(a) = \frac{1}{\gamma(\hat{\ell}_t(a)+\lambda)}, \forall a \in \mathcal{A}$, where λ is a scalar found by binary search such that p_t is a valid distribution;
6: Sample $a_t \sim p_t$; ▷ FTRL
7: Observe feedback y_t;
8: Update $\hat{\mathbf{L}}(a_t)$ and $\mathbf{H}(a_t)$ according to $\mathbf{w}_{t+1}$; ▷ ONS update
9: **end for**

serves as the estimated loss distribution of the current main objective. Specifically, given the construction of such a probability distribution proposed by Abe and Long [2], which describes the probability of arbitrary action a as $p_t(a) = \frac{1}{\gamma(\hat{\ell}_t(a)+\lambda)}$, where γ is the input learning rate, and λ is an unknown parameter such that $\sum_{a=1}^K p_t(a) = 1$ to ensure that p_t is a valid distribution. Notice that $f(\lambda) \coloneqq \sum_{a=1}^K p_t(a) = \sum_{a=1}^K \frac{1}{\gamma(\hat{\ell}_t(a)+\lambda)}$ is strictly monotone for any $\gamma > 0$, the correct parameter λ can be found efficiently by binary search. Once the algorithm obtains the estimated loss distribution on the main objective p_t, it selects an action a_t through random sampling from this distribution. Sampling actions according to such an estimated loss distribution is also known as *follow-the-regularized-leader* (FTRL) with log-barrier regularizer [3].

By this decision-making of choosing the sampled a_t, a loss on the main objective with respect to a_t is incurred, representing whether such a choice is good or not; meanwhile, feedback y_t related to such a loss, as is assumed in (1), is returned to the algorithm. Upon receiving the feedback, the algorithm conducts *online Newton step* (ONS) [16] to update the estimation of the current choice a_t. Specifically, the ONS update involves two steps: the first step is to update the Hessian estimation of the current choice a_t by

$$\mathbf{H}(a_t) \leftarrow \mathbf{H}(a_t) + 4\left(y_t - \hat{\ell}_t(a_t)\right)^2 \mathbf{w}_t \mathbf{w}_t^\top, \tag{5}$$

and the second step is to update the estimated sub-objective vector of the action a_t,

$$\hat{\mathbf{L}}(a_t) \leftarrow \Pi_{[0,1]^d}^{\mathbf{H}(a_t)} \left[\hat{\mathbf{L}}(a_t) + 32\left(y_t - \hat{\ell}_t(a_t)\right)\mathbf{H}(a_t)^{-1}\mathbf{w}_t\right], \tag{6}$$

where $\Pi_{[0,1]^d}^{\mathbf{H}(a_t)}[\cdot] \coloneqq \arg\min_{\mathbf{x}\in[0,1]^d} \sqrt{(\mathbf{x}-\cdot)^\top \mathbf{H}(a_t)(\mathbf{x}-\cdot)}$ is the projection operator onto $[0,1]^d$ in the norm induced by $\mathbf{H}(a_t)$. After updating the estimations with respect to the current choice a_t by ONS, the algorithm steps into the next time and repeat the above estimating-sampling-updating procedures until time T. We present our proposed algorithm in Algorithm 1.

As we can see, the proposed VaRons algorithm is mainly composed of two critical components: ONS and FTRL with log-barrier regularizer. What's interesting about these two components is that they not only handle different difficulties respectively but also are mutually complementary. Specifically, as discussed previously, the main challenges of the LVO problem are three-folded: *non-i.i.d. loss distribution of the main objective* and *partial observability* make it difficult to estimate the main objective accurately, the *dynamic best action* requires an adaptive selection strategy without much dependence on historical choices. Accordingly, as a second-order optimization method that achieves fast rate convergence estimation, ONS treats the non-i.i.d. loss distribution with sufficient convergence rate to track such variations; meanwhile, ONS avoids directly estimating the loss on the main objective. Instead, ONS maintains action-wise loss estimations on all sub-objectives $\hat{\mathbf{L}}$ for each action, according to which ONS can then recover the loss on the main objective based on the linear structure in (1), solving the difficulty posed by partial observability. Meanwhile, FTRL with log-barrier regularizer, samples an action accordingly by constructing a distribution. Since this sampling distribution is adaptive to the loss estimation provided by ONS, the actions sampled by FTRL with log-barrier are also adaptive to the ONS estimations, thus overcoming the difficulty posed by a dynamic best action. However, as a second-order optimization method, ONS requires the exponential-concave condition of the loss function, which our regret definition in (2) and (4) do not satisfy. In light of the analysis in Foster and Rakhlin [12], we state that the randomized sampling strategy of FTRL with log-barrier can provide exponential-concavity for the ONS, which guarantees ONS can be correctly conducted. We put the corresponding theoretical discussion of this statement in Appendix A.

The following theorem ensures that our proposed approach is *minimax optimal* up to logarithmic factors of T.

Theorem 3.1. *Choosing learning rate* $\gamma = O\left(\sqrt{\frac{T}{d\log(T/K)}}\right)$, *Algorithm 1 ensures a cumulative (pseudo) dynamic regret for the general LVO that is upper bounded by*

$$\bar{\mathcal{R}}_T = O\left(K\sqrt{dT\log\left(\frac{T}{K}\right)}\right).$$

Remark 1 (Minimax optimality.). *We construct the lower bound for the general LVO by considering the following special case: setting the weight vector* $\mathbf{w}_t = [1, 0, \ldots, 0]$ *for all* $t \in [T]$, *and thus only the first sub-objective matters and the main objective is exactly the first sub-objective. In this case, the general problem reduces to the traditional* K*-armed bandits problem, whose minimax lower bound is* $\Omega(\sqrt{KT})$ *given by [5, 15]. This means there exists a problem instance (the above instance) that no algorithm can enjoy regret better than our results in terms of total time horizon* T, *and thus, our result is minimax optimal up to logarithmic factors of* T.

Notice that the dependency on the number of actions K in our result is linear and thus sub-optimal. When the number of actions K is large, this could result in additional computational intractability. In the following part, we extend our approach to scenarios where the actions are represented by m-dimensional feature vectors. We demonstrate that, with an additional assumption, the linear dependency of $O(K)$ can be further improved to $O(\sqrt{K})$.

3.2 The contextual variant of LVO

In this part, we extend our learning problem to the contextual scenario, where actions are described with extra information. For example, each individual mode can be treated as a combination of

several parameter settings and states. This parameterized combination can then be treated as the contextual feature of the action.

We formulate the contextual LVO problem where actions are represented by contextual features as follows: we follow the assumptions of linear bandits [1, 2], where each action is represented by an m-dimensional bounded contextual feature $\mathcal{X}(a) \in [0,1]^m$, and each sub-objective can be linearly represented by such contextual features. Formally, for each of the sub-objectives, there exists a parameter $\theta_i \in \mathbb{R}^m$ such that

$$l_i(a) = \theta_i^\top \mathcal{X}(a), \forall i \in [d]. \tag{7}$$

Again, the $l_i(a)$ is the loss on the i-th sub-objective for action a.

Based on the proposed VaRons algorithm, we proposed its variant for the contextual LVO: the ConVaRons algorithm, short for *contextual varied-objective online Newton step*. The algorithm first initializes the inversed learning rate β and the regularization parameter ϵ. Similar to the procedure in VaRons, for each time $t \in [T]$, the algorithm receives the weight vector $\mathbf{w}_t$ and estimates the main objective for each action, and the sampling distribution p_t is calculated, according to which an action is sampled, then the loss on the main objective is incurred with corresponding feedback returned to the learner. The above procedure is almost the same as in Algorithm 1, except that we need a small adaptation when estimating the main objective from the view of sub-objectives: the algorithm first reshapes the estimated parameter, which is a (dm)-dimensional vector, into a $d \times m$ matrix $\hat{\Theta}$, such that $\hat{\Theta}_{i,j} = \hat{\theta}_{(i-1)m+j}$ for $i \in [d], j \in [m]$. The physical intuition of the matrix is obvious: each roll corresponds to a sub-objective, and each column corresponds to the linear parameter in (7). When the feedback $\ell_t(a_t)$ is returned, the algorithm performs the ONS update as follows: the algorithm first constructs the dummy vector $\mathbf{z}_t \in \mathbb{R}^{dm}$ as follows,

$$\mathbf{z}_{t,k} = \mathbf{w}_{t,i}\mathcal{X}_j(a), i = \left\lceil \frac{k}{m} \right\rceil, j = k - m(i-1), k \in [dm], \tag{8}$$

where the footnote regarding i, j, k specifies the corresponding entries in the vector $\mathbf{z}_t$. After transforming the contextual features to a dummy vector, we update corresponding estimations as follows,

$$\mathbf{H} \leftarrow \mathbf{H} + 4\left(y_t - \hat{\theta}^\top \mathbf{z}_t\right)^2 \mathbf{z}_t \mathbf{z}_t^\top, \tag{9}$$

$$\hat{\theta} \leftarrow \Pi^{\mathbf{H}}_{\mathbb{R}^{dm}}\left[\hat{\theta} + \frac{2}{\beta}\left(y_t - \hat{\theta}^\top \mathbf{z}_t\right)\mathbf{H}^{-1}\mathbf{z}_t\right]. \tag{10}$$

With the updated estimation, the algorithm repeats the above estimating-sampling-updating procedure until the time horizon T. We summarize the proposed ConVaRons algorithm in Algorithm 2.

The difficulty within contextual LVO is how to deal with the correlations between actions brought by the contextual structure in (7). Specifically, in the general LVO, we make no assumptions about the relationships within actions, and thus, they are estimated independently, with updating only for the current choice a_t; in contrast, due to the contextual structure, actions within the action set are no longer independent to each other but share the same underlying linear model. In this case, we adopt different ONS estimations of the loss on the main objective. Unlike maintaining action-wise estimations in Algorithm 1 that total K independent estimations are needed, ONS in ConVaRons maintains a single estimation of the underlying linear parameter $\hat{\theta} \in \mathbb{R}^{dm}$, as well as the corresponding

Algorithm 2 ConVaRons: Contextual Varied-objective Online Newton Step

Require: learning rate $\gamma > 0$

initialize: $\beta = \min\{\frac{1}{4}, \frac{d}{16m}\}, \epsilon = \frac{d}{\beta^2 m}, \hat{\theta} = \mathbf{0}^{dm}, \mathbf{H} = \epsilon I_{dm}$ ▷ estimations and Hessians

for $t = 1, \cdots, T$ **do**

  Receive weight vector $\mathbf{w}_t$;

  Reshape the estimation $\hat{\theta}$ into a d by m matrix $\hat{\Theta} \in \mathbb{R}^{d \times m}$, such that $\hat{\Theta}_{i,j} = \hat{\theta}_{(i-1)m+j}$ for $i \in [d], j \in [m]$

  Estimate $\hat{\ell}_t(a) = \mathbf{w}_t^\top \hat{\Theta} \mathcal{X}(a), \forall a \in \mathcal{A}$;

  Compute $p_t(a) = \frac{1}{\gamma(\hat{\ell}_t(a) + \lambda)}, \forall a \in \mathcal{A}$, where λ is a scalar found by binary search such that p_t is a valid distribution, sample $a_t \sim p_t$; ▷ FTRL

  Observe feedback y_t;

  Calculate the dummy vector $\mathbf{z}_t$ according to (8)

  Update according to (9) and (10); ▷ ONS update

end for

Hessian estimation $H \in \mathbb{R}^{dm \times dm}$. By combining all estimations into the same parameter, selecting any action provides information about the underlying linear parameters.

Formally speaking, the following theorem states that the $O(K)$ dependence of Algorithm 1 can be improved by to $O(\sqrt{mK})$ by Algorithm 2, and meanwhile preserves the minimax optimality,

Theorem 3.2. *Choosing optimal learning rate $\gamma = O\left(\sqrt{\frac{KT}{dm \log T}}\right)$, Algorithm 2 ensures a cumulative (pseudo) dynamic regret for the contextual LVO that is upper bounded by*

$$\bar{\mathcal{R}}_T = O\left(\sqrt{dmKT \log T}\right).$$

Remark 2 (Minimax optimality). *The sub-optimal linear dependency of $O(K)$ for Algorithm 1 is improved to $O\left(\sqrt{K}\right)$. Meanwhile, as for the T dependent term, similar to what we do in the general case, we can construct the special case by setting $\mathbf{W}_t = [1, 0, \ldots, 0]$ for all $t \in [T]$, and thus only the first sub-objective matters and the main objective is exactly the first sub-objective. In this case, the general problem reduces to the stochastic linear bandits with finitely many arms, whose minimax lower bound is $\Omega(\sqrt{mT \log(T/m)})$ given by [22]. This implies the ConVaRons algorithm preserves the minimax optimality in terms of T up to logarithmic factors.*

4 Experiments

In this section, we conduct experiments on two applications of the proposed LVO problem, estimating the efficacy of our approaches.

4.1 Dynamic classifier selection

Classifiers optimal for one objective may be sub-optimal for another [45], necessitating adaptive selection as data characteristics and performance metrics change, a process known as dynamic classifier selection (DCS) [17, 44]. DCS is an ensemble method where, after training multiple classifiers, one or a combination of classifiers is dynamically selected for test instances. In our experiments, we simplify DCS by selecting one classifier at a time, modeling it as an

example of the LVO problem. This approach has many real-world applications. For instance, in an autonomous driving system (Figure 1), different modes can be seen as trained classifiers performing variably across metrics. On simple roads, accuracy and F1 score are primary, while on complex roads, the End-to-End Lateral Deviation (E2E-LD) metric is more suitable [30].

In this part, we conduct experiments on an application of the LVO problem: the dynamic classifier selecting tasks, where data come with a stream with varied performance metrics.

4.1.1 Selecting Non-contextual Classifiers. In this part, we conduct classifier selection tasks on non-contextual classifiers.

Basic Settings. We conduct the experiment on the IJCNN1 dataset [8]. Specifically, we first train 10 Core Vector Machine classifiers [34] on the training data. These classifiers are served as the action set of online decision-making problem. For each classifier, the evaluation metrics include accuracy, f1-score, Area Under the ROC Curve (AUC), and average precision (AP), which are taken as the sub-objectives. The main objective combines these sub-objectives into a composite measure. For each time step $t = 1, \ldots, 1000$, we generate a weight vector to represent potential variations in these sub-objectives. Upon receiving the weight vector, the algorithm selects one classifier from the 10 trained classifiers. Then, we randomly sample 5% of the test data to simulate the mini-batch in streaming data for evaluation. The selected classifier is tested on the mini-batch and incurs the loss on the main objective, which is then returned to the algorithm for future selections. The goal is to minimize the loss concerning the main objective.

Simulated weights variations. We establish the following two variation environments for the weights in the main objective:

- Drift: the environment simulates the cases when the weight vector varies smoothly. Specifically, the weight vector initially starts at $\mathbf{w}_t = [1, 0, 0, 0]$ and gradually transitions to the uniform distribution and then to $\mathbf{w}_t = [0, 1, 0, 0]$. The procedure continues for the third and fourth sub-objectives for several times till end.
- Random: the environment simulates user-specific tasks, with each user having distinct requirements for the classifier. Consequently, the weight vector at each time is randomly generated in a 4-dimensional simplex.

To quantify the variation of the main objective, we define the weight path length as $\mathcal{P}_T := \sum_{t=1}^{T-1} \|\mathbf{w}_{t+1} - \mathbf{w}_t\|_2^2$. The weight path length for drift $\mathcal{P}_T = 14$ represents a mild variation; the weight path length for Random $\mathcal{P}_T = 470$ represents a more drastic variation.

Contenders. We compare the VaRons algorithm with the following state-of-the-art decision-making algorithms in the most related topics; the specific reason for such choosing is given in Appendix B. Parameters are set optimally according to theoretical guidance.

- Ground truth Pareto front (GT-PF). This method randomly chooses one action from the ground truth Pareto front. Notice that MOO methods searching for the Pareto front will always converge to the ground truth result, and thus, this approach includes the best MOO algorithms by assuming they successfully identify the Pareto front at the beginning.
- EXP3.S [5]. The EXP3.S algorithm handles adversarial rewards, which makes it possible to handle decision-making with time-varying bandit feedback on the main objective.

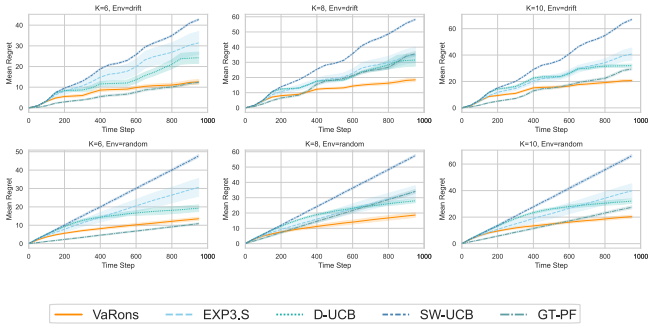


Figure 3: Cumulative regret results on the non-contextual dynamic classifier selection task, the error presents the standard deviation region. The lower the results, the better. We conduct for $K = 6, 8, 10$ actions under both 'drift' and 'random' environmental settings. Our algorithm, VaRons, exhibits lower regret compared to contenders in most cases.

- SW-UCB [14] and D-UCB [20]. The SW-UCB and D-UCB handle non-stationary stochastic bandit feedback, and thus is also possible to handle time-varying main objective.

Results. We present the average cumulative regret results in Figure 3, the results are averaged over 20 independent trials. As is presented, VaRons obtains the lowest cumulative regrets in most scenarios; compared with the GT-PF when the number of actions K is small when the cardinality of the Pareto front is small, GT-PF outperforms our approach, it is emphasized that such results are caused by the selection of too small time horizon T, as regrets incurred by GT-PF are always proportion to T with small scope, regrets incurred by VaRons is converging and thus is sub-linear in T. Meanwhile, VaRons has similar performance when the variation environment is different, implying a regret that is *irrelevant* to the extent of variation, matching the theoretical findings in Theorem 3.1.

4.1.2 Selecting Parameterized Classifiers. In this part, we focused on the parameterized classifier selection, where classifiers are parameterized and described as an application for the contextual LVO.

Data and task description. We select classifiers, which are parameterized represented, on the SNW data set [48] and the NoC data set [4]. The SNW dataset contains 206 different designs synthesized for a Field-Programmable Gate Array (FPGA) platform, each characterized by a 4-dimensional contextual representation. The sub-objectives are two conflicting aspects of chip designs: *area* and *throughput*. The NOC dataset contains 259 different NoC implementations, each represented by a 5-dimensional contextual representation. The sub-objectives of these implementations are *energy consumption* and *running time*. Similarly, we generated weights at each time to simulate possible variation scenarios.

Simulated weights variations Similarly, we set 2 variation environments: the drift scenario with weight path length $\mathcal{P}_T = 30$, and the random scenario with weight path length $\mathcal{P}_T = 470$, representing mild and drastic variation environments, respectively. Meanwhile, we randomly select $K = 10, 20, 40$ designs as the action sets, each represented by its corresponding contextual features.

Contenders. We compare the proposed ConVaRons algorithm (Algorithm 2) with the following state-of-the-art algorithms of the most related topics; all parameters are set optimally according to

Table 2: Cumulative regret results on the SNW dataset of the contextual dynamic classifier selection tasks in mean ± std (rank) format, the smaller the results and ranks, the better.

Scenarios	ConVaRons (OURS)	*Contenders*			
		GT-PF	SW-LinUCB	D-LinUCB	RestartUCB
10D	**13.71±0.76** (1)	18.29±0.91 (2)	24.66±1.18 (3)	34.83±5.24 (5)	28.25±0.92 (4)
10R	**13.62±0.92** (1)	18.09±0.78 (2)	25.15±0.95 (4)	20.33±1.77 (3)	29.41±1.25 (5)
20D	**14.01±0.77** (1)	23.70±0.46 (2)	34.72±1.17 (3)	45.85±6.17 (5)	34.81±1.46 (4)
20R	**17.32±1.99** (1)	23.77±0.55 (2)	35.51±1.53 (4)	26.39±1.55 (3)	35.74±1.87 (5)
40D	**16.47±1.44** (1)	32.98±0.29 (2)	46.05±1.07 (4)	52.82±4.96 (5)	43.98±1.34 (3)
40R	**21.52±2.66** (1)	32.99±0.34 (2)	46.75±1.39 (5)	35.42±2.60 (3)	46.48±1.95 (4)
Avg. Rank	**1.00**	2.00	3.83	4.00	4.17

Table 3: Cumulative regret results on the NOC dataset of the contextual dynamic classifier tasks in mean ± std (rank) format, the smaller the results and ranks, the better.

Scenarios	ConVaRons (OURS)	*Contenders*			
		GT-PF	SW-LinUCB	D-LinUCB	RestartUCB
10D	**21.68±1.04** (1)	23.15±0.99 (2)	41.10±1.76 (4)	49.20±3.54 (5)	39.37±1.48 (3)
10R	23.99±1.31 (2)	**22.69±0.83 (1)**	43.42±1.49 (5)	35.92±1.99 (3)	41.78±1.53 (4)
20D	**30.28±1.67** (1)	33.42±0.60 (2)	46.69±1.56 (4)	48.13±3.30 (5)	43.64±1.13 (3)
20R	33.67±1.34 (2)	**33.20±1.48 (1)**	49.65±1.88 (5)	41.35±1.65 (3)	48.73±1.33 (4)
40D	**41.08±3.03** (1)	41.84±0.63 (2)	47.65±1.31 (5)	45.95±3.20 (4)	43.59±1.05 (3)
40R	44.03±1.52 (2)	**41.81±1.79 (1)**	53.30±1.80 (5)	45.56±1.51 (3)	51.21±1.40 (4)
Avg. Rank	**1.50**	**1.50**	4.67	3.83	3.50

theoretical guidance or grid search on validation data. Similarly, we give the reason of choosing these contenders is given in Appendix B,

- GT-PF. This approach selects one of the ground truth Pareto front randomly, representing the MOO-type of methods.
- SW-LinUCB [9], D-LinUCB [28] and RestartUCB [41]. All handle the non-stationary stochastic linear bandits, implying the potential to handle the changing main objectives when actions are represented by contextual features.

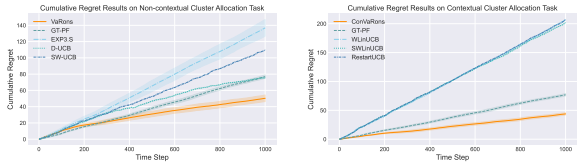
Results. The cumulative results on the two tasks are presented in Table 2 and Table 3 in mean ± std (rank) format. Similarly, the results are averaged on 20 independent trials. Again, GT-PF outperforms our approach in some tasks in the NOC dataset. We explain this by the existence of noisy data such that the linear assumption in (7) does not entirely hold in this dataset, and thus approximating sub-objectives with this linear structure introduces extra inaccuracy in estimations. However, our proposed ConVaRons still obtains the best (or tied for best) averaged performance over all tasks.

4.2 Cluster service allocation

In this part, we focus on large-scale cluster service allocation tasks.

4.2.1 Task description. We conduct experiments on the online service cluster trace data from the Alibaba Cluster Trace Program,[2] which contains $1.4M$ users' requirements on online service in time sequence. The task focuses on designing algorithms for cluster service providers to allocate computational tasks to different machines. Specifically, the service provider first categorizes available machines into several clusters according to configurations, each has different performance on processing abilities, memory size, and storage capabilities. We take each cluster as an optional action in the action set. With the timestamp, users continuously submit their computational tasks to the service provider, each with different requirements for CPU, memory, and disk space, which we take as the sub-objectives. For each submitted task, the service provider first obtains the weights for sub-objectives according to the specific requirements and then allocates this task to the most suitable cluster. The main objective is to maximize the overall Quality of Service (QoS), which is assumed to be a weighted combination of sub-objectives as different clusters may prioritize different sub-objectives, resulting in different overall QoS. We note that the cluster service allocation task is also related to the recently proposed new paradigm called *CoRE-Learning* [46], which takes the computational resource into account of machine learning theory.

[2]https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2017/trace_201708.md



(a) Non-contextual case (b) Contextual case

Figure 4: Cumulative regret results on the dynamic cluster service allocation task, the error presents the standard deviation region. A lower regret indicates a better allocation strategy. Notice that the SW-UCB and D-UCB algorithms are deterministic strategies decided by user requests in the records; there are no error regions for these two algorithms.

4.2.2 Constructing the LVO problem. In this part, we introduce how we process available data to construct an LVO problem.

Assigning weights for sub-objectives. We allocate weights based on user-defined requirements for CPU, memory, and disk space according to the Bradley–Terry model [7]. Specifically, we rescale these requests to standard distributions. Then, we assign weights in proportion to the exponential of their scaled values. This approach ensures a sensitive weight distribution, reflecting the relative significance of each requirement. For instance, the weight for CPU is assigned as:

$$\mathbf{w}_t(\text{CPU}) = \frac{\exp(\text{scaled sub-objective 1 requirement})}{\sum_{i=1}^{3} \exp(\text{scaled sub-objective } i \text{ requirement})}.$$

Constructing the Decision-Making Framework. To avoid mismatches between the algorithm's selections and historical allocations, we follow the evaluation method of Li et al. [21], Mehrotra et al. [24]. This method validates a task only if the selected cluster by the algorithm aligns with its historical allocation. Non-valid tasks are skipped until $T = 1000$ tasks are reached. For each valid match, we take QoS as the returned feedback y_t. Specifically, the QoS is categorized into five levels: Excellent (Ex), Satisfactory (Sa), Standard (St), Substandard (Su), and Unsatisfactory (Un), corresponding to loss intervals of $[0, 0.2]$, $[0.2, 0.4]$, $[0.4, 0.6]$, $[0.6, 0.8]$,$[0.8, 1]$, respectively. For analytical convenience, we convert these levels into scalar losses at the midpoint of intervals: 0.1, 0.3, 0.5, 0.7,0.9.

The contextual LVO settings. For each cluster, we can characterize it with some performance metrics, such as cycles per instruction (CPI), misses per thousand instructions (MPKI), and real-time disk load that represent the execution time, cache usage, and response times, respectively. We take these critical states as the cluster's contextual feature and additionally conduct contextual LVO experiments on the proposed ConVaRons algorithm.

Contenders. Similar to the experiments in Section 4.1, we select EXP3.S [5], SW-UCB [14], D-UCB [20] and GT-PF as the contenders for VaRons in the general LVO, and SW-LinUCB [9], D-LinUCB [28], RestartUCB [41] and GT-PF as the contenders for ConVaRons in the contextual LVO.

4.2.3 Results. We present the cumulative regret results in Figure 4 and additionally illustrate the loss of each user's request in Figure 5. For better illustration, we combined the results of other contenders except GT-PF. As the loss for each user request represented by the shadow is too noisy, we additionally draw the smoothed average loss, that averaged on the past 50 user requests, as well as the overall averaged loss. In both cases, our proposed algorithm presents lower losses than contenders, indicating a better allocation strategy with higher adaptivity to each user's specific requirements.

5 Related work

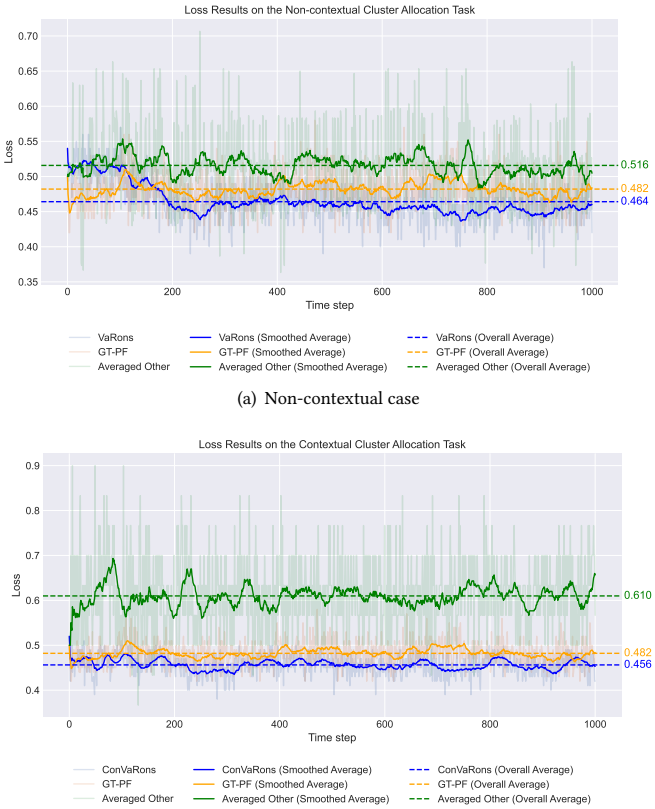
The LVO problem shares characteristics with multi-dimensional and non-stationary losses. Here, we briefly discuss two related areas and explain why existing algorithms cannot directly address the LVO problem. Detailed discussions are presented in Appendix B.

The LVO problem arises in open-environment machine learning (open ML) [45], which fundamentally differs from conventional static machine learning. Open ML faces unique challenges such as the emergence of new classes [10, 38, 42], evolving features [18, 39], shifting data distributions [31, 40], and varied learning objectives. Handling varied objectives is a crucial yet underexplored problem within this context. As objectives shift with specific requirements, open environments necessitate machine learning models that are highly flexible and adaptable. Effectively handling varied objectives is fundamental to addressing practical problems in open ML.

One related area is multi-objective decision-making methods, particularly *multi-objective multi-armed bandits* [11, 26, 32], which focus on actions with multi-dimensional losses. Drugan and Nowé [11] proposed both Pareto and scalarized optimality settings. While most subsequent works [6, 25] focus on Pareto optimality, scalarized optimality, which combines sub-objectives using fixed weights, is more relevant to our problem. However, as the weights are assumed to be fixed, it is not suitable to handle changing weights.

Another relevant area is decision-making based on non-stationary losses. This approach treats the main objective as a non-stationary target, with action-wise loss changing based on varying weight vectors. The LVO problem can be viewed as a special case of non-stationary decision-making, where each action corresponds to a non-stationary reward or loss. Algorithms like Exp3.S [5] guarantee an optimal $O(\sqrt{S_T T})$ regret bound, where S_T is the number of distribution shifts by time T. Subsequent algorithms, such as D-UCB and SW-UCB [14, 20], share this optimality.

For the contextual extension of LVO, the linear structure assumption is similar to those in stochastic linear bandits [1, 2], especially non-stationary stochastic linear bandits [9, 36]. Algorithms like SW-LinUCB [9], D-LinUCB [28], and RestartUCB [41] share sub-optimal $O(P_T^{\frac{1}{4}} T^{\frac{3}{4}})$ regret, where P_T measures environmental changes. However, $S_T = \Theta(T)$ and $P_T = \Theta(T)$ in the worst case, the regret bound is linearly dependent on T, implying that selections do not converge to the best action. Hence, handling LVO with reduction to non-stationary linear bandits are not suitable.



(a) Non-contextual case

(b) Contextual case

Figure 5: Loss according to QoS of VaRons algorithm against GT-PF and an average of other contenders on the cluster allocation tasks. Lower loss represents better allocation strategy. In the non-contextual case, VaRons shows consistent loss minimization, closely followed by GT-PF, with the overall average loss for each algorithm indicated by dashed lines; for the contextual case, ConVaRons and GT-PF exhibit similar trends, with VaRons maintaining a lower loss profile as evidenced by the overall average.

6 Conclusion and Future work

We present a formulation of two settings within the problem of Learning with Varied Objectives (LVO): the general LVO, which lacks contextual representation for each action, and the contextual LVO, where actions are characterized by contextual features. For these respective settings, we introduce the VaRons and ConVaRons algorithms, both of which are provably minimax optimal in terms of T, up to logarithmic factors. Empirically, we validate the efficacy of our approaches through dynamic classifier selection and cluster service allocation tasks, providing empirical evidence for our theoretical findings. As discussed in the experiment part, our proposed algorithms for LVO may be useful for CoRE-learning [19, 46], particularly when considering varying objectives during resource allocation. We leave this for future investigation.

Acknowledgement

We acknowledge the funding provided by the National Science Foundation of China (61921006, 62206245, 62206125) and Collaborative Innovation Center of Novel Software Technology and Industrialization. Z.-H. Zhou is the corresponding author.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 2312–2320, 2011.

[2] Naoki Abe and Philip M. Long. Associative reinforcement learning using linear probabilistic concepts. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 3–11, 1999.

[3] Amit Agarwal and Elad Hazan. Efficient algorithms for online game playing and universal portfolio management. *Electronic Colloquium on Computational Complexity*, TR06-033, 2006.

[4] Oscar Almer, Nigel Topham, and Björn Franke. A learning-based approach to the automated design of mpsoc networks. In *Proceedings of the 24th IEEE International Conference on Architecture of Computing Systems (ARCS)*, 243–258, 2011.

[5] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1): 48–77, 2002.

[6] Peter Auer, Chao-Kai Chiang, Ronald Ortner, and Madalina M. Drugan. Pareto front identification from stochastic bandit feedback. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 939–947, 2016.

[7] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[8] Chih-Chung Chang and Chih-Jen Lin. Ijcnn 2001 challenge: Generalization ability and text decoding. In *Proceedings of the 10th International Joint Conference on Neural Networks (IJCNN)*, 1031–1036, 2001.

[9] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1079–1087, 2019.

[10] Qing Da, Yang Yu, and Zhi-Hua Zhou. Learning with augmented class by exploiting unlabeled data. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, 1760–1766, 2014.

[11] Madalina M. Drugan and Ann Nowé. Designing multi-objective multi-armed bandits algorithms: A study. In *Proceedings of the 22nd International Joint Conference on Neural Networks (IJCNN)*, 1–8, 2013.

[12] Dylan J. Foster and Alexander Rakhlin. Beyond UCB: optimal and efficient contextual bandits with regression oracles. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 3199–3210, 2020.

[13] Alex A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *ACM SIGKDD Explorations*, 6(2):77–86, 2004.

[14] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Proceedings of the 22nd International conference on algorithmic learning theory (ALT)*, 174–188, 2011.

[15] Sébastien Gerchinovitz and Tor Lattimore. Refined lower bounds for adversarial bandits. In *Advances in Neural Information Processing Systems 29 (NIPS)*, 1190–1198, 2016.

[16] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT)*, 499–513, 2006.

[17] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[18] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. Learning with feature evolvable streams. *IEEE Transactions on Knowledge and Data Engineering*,33(6): 2602–2615, 2021.

[19] Jing Wang, Miao Yu, Peng Zhao, and Zhi-Hua Zhou. Learning with Adaptive Resource Allocation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, to appear, 2024.

[20] Levente Kocsis and Csaba Szepesvári. Discounted ucb. In *2nd PASCAL Challenges Workshop*, 2: 51–134, 2006.

[21] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the 4th ACM International conference on Web Search and Data Mining (WSDM)*, 297–306, 2011.

[22] Yingkai Li, Yining Wang, and Yuan Zhou. Nearly minimax-optimal regret for linearly parameterized bandits. In *Proceedings of the 33rd Conference on Learning Theory (COLT)*, 2173–2174, 2019.

[23] Marler R. Timothy and Jasbir S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[24] Rishabh Mehrotra, Niannan Xue, and Mounia Lalmas. Bandit based optimization of multiple objectives on a music streaming platform. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 3224–3233, 2020.

[25] Kristof Van Moffaert, Kevin Van Vaerenbergh, Peter Vrancx, and Ann Nowé. Multi-objective χ-armed bandits. In *Proceedings of the 23rd International Joint Conference on Neural Networks (IJCNN)*, 2331–2338, 2014.

[26] Tadahiko Murata, Hisao Ishibuchi, and Hideo Tanaka. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957–968, 1996.

[27] Diederik M. Roijers, Luisa M. Zintgraf, and Ann Nowé. Interactive thompson sampling for multi-objective multi-armed bandits. In *Proceedings of the 5th International Conference on Algorithmic Decision Theory (ADT)*, 18–34, 2017.

[28] Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted linear bandits for non-stationary environments. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*,12017–12026, 2019.

[29] Ilya O. Ryzhov, Warren B. Powell, and Peter I Frazier. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

[30] Takami Sato and Qi Alfred Chen. Towards driving-oriented metric for lane detection models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 17153–17162, 2022.

[31] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments - Introduction to Covariate Shift Adaptation*. MIT Press, 2012.

[32] Cem Tekin and Eralp Turğay. Multi-objective contextual multi-armed bandit with a dominant objective. *IEEE Transactions on Signal Processing*, 66(14):3799–3813, 2018.

[33] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[34] Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung, and Nello Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(4):363–392, 2005.

[35] Christian Von Lücken, Benjamín Barán, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58:707–756, 2014.

[36] Chen-Yu Wei and Haipeng Luo. Non-stationary reinforcement learning without prior knowledge: an optimal black-box approach. In *Proceedings of the 34th International Conference on Learning Theory (COLT)*, 4300–4354, 2021.

[37] Saba Q. Yahyaa and Bernard Manderick. Thompson sampling for multi-objective multi-armed bandits problem. In *Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN)*, 47–52,2015.

[38] Yu-Jie Zhang, Peng Zhao, Lanjihong Ma, and Zhi-Hua Zhou. An unbiased risk estimator for learning with augmented classes. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 10247-10258, 2020.

[39] Zhen-Yu Zhang, Peng Zhao, Yuan Jiang, and Zhi-Hua Zhou. Learning with feature and distribution evolvable streams. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 11317–11327, 2020.

[40] Peng Zhao, Guanghui Wang, Lijun Zhang, and Zhi-Hua Zhou. Bandit convex optimization in non-stationary environments. In *Journal of Machine Learning Research*, 22(125): 1-45, 2021.

[41] Peng Zhao, Lijun Zhang, Yuan Jiang, and Zhi-Hua Zhou. A simple approach for non-stationary linear bandits. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 746–755, 2020.

[42] Peng Zhao, Jia-Wei Shan, Yu-Jie Zhang, and Zhi-Hua Zhou. Exploratory machine learning with unknown unknowns. *Artificial Intelligence*, 327:104059, 2024.

[43] Yong Zheng and David Wang. Multi-objective recommendations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, 4098–4099, 2021.

[44] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[45] Zhi-Hua Zhou. Open-environment machine learning. *National Science Review*, 9 (8):nwac123, 2022.

[46] Zhi-Hua Zhou. Learnability with time-sharing computational resource concerns. *National Science Review*, nwae204, 2024.

[47] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 928–936, 2003.

[48] Marcela Zuluaga, Peter A. Milder, and Markus Püschel. Computer generation of streaming sorting networks. In *Proceddings of the 49th Annual Design Automation Conference (DAC)*, 1245–1253, 2012.

Supplementary Materials

In the appendix, we provide theoretical analysis in Appendix A, and provide the full content of related works in Appendix B.

A Theoretical Analysis

In this section, we provide the theoretical proofs for the main theorems: Theorem 3.1 for VaRons and Theorem 3.2 for ConVaRons.

A.1 Proof of Theorem 3.1

We first focus on Theorem 3.1 for VaRons in the general Learning with Varied objectives (LVO) problem. As mentioned in the main text, our algorithm comprises two key components: Follow-the-Regularized-Leader (FTRL) with a log-barrier regularizer, and the Online Newton Step (ONS). The following lemma states that the cumulative (pseudo) regret defined in (4) can be upper bounded by the square loss of ONS, plus an additional constant term.

Lemma A.1. *Sampling actions according to $p_t(a) = \frac{1}{\gamma(\hat{\ell}_t(a)+\lambda)}$ (FTRL with log-barrier) guarantees an upper bound on the cumulative pseudo regret of:*

$$\begin{aligned}\bar{\mathcal{R}}_T &\le \frac{\gamma}{4}\sum_{t=1}^{T}\mathbb{E}\left[(\hat{\ell}_t(a_t)-y_t)^2-(\ell_t(a_t)-y_t)^2\right]+\frac{(K-1)T}{\gamma}\\ &= \frac{\gamma}{4}\sum_{a\in\mathcal{A}}\underbrace{\sum_{t:a_t=a}\mathbb{E}\left[(\hat{\ell}_t(a)-y_t)^2-(\ell_t(a_t)-y_t)^2\right]}_{\textit{Term A}}+\frac{(K-1)T}{\gamma},\end{aligned}$$

where $\ell_t(a) = \mathbf{w}_t^\top \mathbf{L}(a)$ and $\hat{\ell}_t(a) = \mathbf{w}_t^\top \hat{\mathbf{L}}(a)$ are the main objective and its estimation for action a according to line 4 of VaRons (Algorithm 1).

As shown, the effectiveness of FTRL with a log-barrier regularizer is two-fold. Firstly, the cumulative (pseudo) regret is upper bounded by the estimation error on the square loss (Term A), with additional constant term. If Term A satisfies the exponential-concavity condition for ONS, it can be considered a surrogate loss function for ONS. Secondly, the right side of the inequality eliminates the term related to the dynamic competitor a_t^*, defined in (3), thereby addressing the technical difficulties posed by the dynamic competitor.

We now focus on Term A. The following lemma guarantees the exponential-concavity condition of Term A.

Lemma A.2. *For any $\mathbf{x} \in [0,1]^d$ and $\mathbf{w}_t \in \Delta_{d-1}$, the loss function defined by $f(\mathbf{x}) := (\mathbf{w}_t^\top \mathbf{x} - y_t)^2$ is a $\frac{1}{2}$-exponential-concave function with $\|\nabla f(\mathbf{x})\|_2 \le \frac{2}{\sqrt{d}}$. Furthermore, for any $\mathbf{y} \in [0,1]^d$, the following holds:*

$$f(\mathbf{x}) \ge f(\mathbf{y}) + \nabla f(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) + \frac{1}{32}(\nabla f(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}))^2.$$

Once Term A fulfills the exponential-concave requirements, we are now able to estimate the action-wise sub-objective vector $\mathbf{L}(a)$ for all $a \in \mathcal{A}$ independently. Recall that $\mathbf{L}(a) = [l_1(a), \dots, l_d(a)]$ is a vectorized representation of the losses on sub-objectives of action a, for an action $a \in \mathcal{A}$, the following bound holds according to the analysis in Hazan et al. [16],

Lemma A.3. *Denote by $\mathcal{I}(a)$ as the set of time when $a_t = a$. The Online Newton Step (line 7 in algorithm 1) guarantees the following upper bound for Term A,*

$$\textit{Term A} \le 8d\log\left(\frac{|\mathcal{I}(a)|}{64}+1\right)+8$$

With the lemmas above, it is sufficient to analyze Theorem 3.1.

Proof of Theorem 3.1. According to Lemma A.1 and Lemma A.3, the cumulative pseudo regret is upper bounded by:

$$\begin{aligned}\bar{\mathcal{R}}_T &\le \frac{\gamma}{4}\sum_{a\in\mathcal{A}}\left(8d\log\left(\frac{|\mathcal{I}(a)|}{64}+1\right)+8\right)+\frac{(K-1)T}{\gamma}\\ &= \frac{\gamma}{4}\left(8d\log\left(\prod_{a\in\mathcal{A}}\frac{|\mathcal{I}(a)|+64}{64}\right)+8K\right)+\frac{(K-1)T}{\gamma}\\ &\le \frac{\gamma}{4}\left(8dK\log\left(\frac{T}{64K}+1\right)+8K\right)+\frac{(K-1)T}{\gamma}.\end{aligned}$$

The last inequality utilizes the AM-GM inequality, and use the fact that $\prod_{a\in\mathcal{A}}|\mathcal{I}(a)| \le \left(\frac{\sum_{a\in\mathcal{A}}|\mathcal{I}(a)|}{K}\right)^K$ and $\sum_{a\in\mathcal{A}}|\mathcal{I}(a)| = T$ by definition. Thus, $\prod_{a\in\mathcal{A}}(|\mathcal{I}(a)|+64) \le \left(\frac{T+64K}{K}\right)^K$. Finally, by setting the optimal learning rate $\gamma = \sqrt{\frac{(K-1)T}{2dK\log\left(\frac{T}{64K}+1\right)+2K}}$, we have:

$$\bar{\mathcal{R}}_T \le 2K\sqrt{2dT\log\left(\frac{T}{64K}+1\right)+2T} = O\left(K\sqrt{dT\log\left(\frac{T}{K}\right)}\right).$$

□

A.2 Proof of Theorem 2

The reshaping step in line 4 of ConVaRons (Algorithm 2) and constructing the dummy vector $\mathbf{z}_t$ as in (8) ensures that $\hat{\ell}_t(a) = \mathbf{w}_t^\top \hat{\Theta}\mathcal{X}(a) = \hat{\theta}^\top \mathbf{z}_t$, and thus $f(\Theta) = (\mathbf{w}_t^\top \Theta\mathcal{X} - y_t)^2$ corresponds to $g(\theta) = (\mathbf{z}_t^\top \theta - y_t)^2$. Therefore, one can replace $\mathbf{w}_t$ and θ in Lemma A.2 with $\mathbf{z}_t$ and θ. Hence, most proofs are based on the analysis of Theorem 1. We have the following lemma, which is an adaptation of the original Lemma A.1, Lemma A.2, and Lemma A.3 in the contextual LVO.

Lemma A.4. *Sampling actions according to $p_t(a) = \frac{1}{\gamma(\hat{\ell}_t(a)+\lambda)}$ (FTRL with log-barrier) guarantees an upper bound on the cumulative pseudo regret of:*

$$\bar{\mathcal{R}}_T \le \underbrace{\frac{\gamma}{4}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}}\mathbb{E}\left[(\hat{\ell}_t(a)-y_t)^2-(\ell_t(a_t)-y_t)^2\right]}_{\textit{Term E}}+\frac{(K-1)T}{\gamma},$$

where $\ell_t(a) = \mathbf{w}_t^\top \Theta\mathcal{X}(a) = \theta^\top \mathbf{z}_t$ and $\hat{\ell}_t(a) = \mathbf{w}_t^\top \hat{\Theta}\mathcal{X}(a) = \hat{\theta}^\top \mathbf{z}_t$ are the main objective and its estimation for action a according to line 5 of ConVaRons (Algorithm 2).

Similarly, when it comes to the conVaRons (algorithm 2), the loss function shares the same $\frac{1}{2}$-exponential-concavity as in Lemma A.2. Formally, we conclude the results as the following lemma:

Lemma A.5. *For any $\theta \in [0,1]^{dm}$, $\mathbf{z}_t$ constructed as in (8), where $\mathbf{w}_t \in \Delta_{d-1}$ and $\mathcal{X} \in [0,1]^m$, the loss function defined by $f(\theta) := (\mathbf{z}_t^\top \theta - y_t)^2$ is a $\frac{1}{2}$-exponential-concave function with $\|\nabla f(\theta)\|_2 \leq 2\sqrt{\frac{m}{d}}$. Further, for any $\eta \in [0,1]^{dm}$, taking $\beta = \min\left\{\frac{1}{4}, \frac{d}{16m}\right\}$ yields the following inequality:*

$$f(\theta) \geq f(\eta) + \nabla f(\eta)^\top(\theta - \eta) + \frac{\beta}{2}(\nabla f(\eta)^\top(\theta - \eta))^2.$$

Lemma A.6. *The Online Newton Step (line 9 in Algorithm 2) guarantees the following upper bound for Term E:*

$$\textit{Term } E \leq 8dm \log\left(\frac{T}{64} + 1\right) + 8.$$

Notice that the action-wise analysis in Lemma A.3 is based on the ONS estimation being action-specific. When applying this to ConVaRons, we replace the K estimations in VaRons with a single estimation for the entire action set $\mathcal{A}$, treating $\mathcal{A}$ as a super-action. This approach still holds since Lemma A.3 focuses solely on estimations, not on the decision-making or sampling process. Therefore, if Lemma A.3 holds for any arbitrary action a over the time interval $\mathcal{I}(\mathcal{A})$, it should also hold for the super-action $\mathcal{A}$. With $n_T = |\mathcal{I}(\mathcal{A})| = T$, $\|\nabla f(\theta)\|_2 \leq 2\sqrt{\frac{m}{d}}$ and $\hat{\theta} \in [0,1]^{dm}$ (different from the estimation in Lemma A.3 where $\hat{\mathbf{L}} \in [0,1]^d$), the analysis is almost the same, we omit this part.

With above lemmas, the proof of Theorem 3.2 follows:

Proof of Theorem 3.2. Combining the results in Lemma A.4 and Lemma A.6, we have:

$$\bar{\mathcal{R}}_T \leq \frac{\gamma}{4} \cdot 8dm \log\left(\frac{T}{64} + 1\right) + 8 + \frac{(K-1)T}{\gamma}.$$

Selecting the optimal $\gamma = \sqrt{\frac{(K-1)T}{2dm\log\left(\frac{T}{64}+1\right)}}$, we have:

$$\bar{\mathcal{R}}_T \leq 2\sqrt{2dm(K-1)T \log\left(\frac{T}{64} + 1\right)} + 8 = O\left(\sqrt{dmKT \log T}\right).$$

□

B Full Related Works

As our problem shares the characteristic of varying objective including several sub-objectives and constantly changing weight vectors that defines the main objective, we separate possible related decision-making approaches that may help to solve the LVO problem into two categories: the multi-objective type of methods, and the non-stationary type of methods.

We first focus on the **multi-objective type of methods**. Decision-making approach that handles multiple objectives is the *Multi-objective multi-armed bandits* (MO-MAB). As each action corresponds to a multi-dimensional reward, LVO is closely related to the MO-MAB problem, formally proposed by Drugan and Nowé [11]. Drugan and Nowé [11] propose both the Pareto and scalarized optimality settings. While most subsequent works [6, 25] focus on the Pareto optimality setting, where the goal is to find the Pareto optimal set of actions, the scalarized optimality setting, which assumes the main objective can be represented by the weighted combination of sub-objectives, is more related to our problem. The only difference is that LVO problem considers time-varying weights. To handle the scalarized MO-MAB problem, Drugan and Nowé [11] proposed the Scalarized MO-UCB algorithm that finds the scalarized optimal action, Yahyaa and Manderick [37] propose the LS-KG and SMOMAB algorithms that utilize knowledge gradient policy [29] and Thompson sampling [33] to handle linear scalarized MO-MAB. These works assume the scalarization function is known in advance, based on which Roijers et al. [27] propose the ITS algorithm that learns the scalarization function by interacting with users. However, to the best of our knowledge, *no prior work addresses a time-varying scalarization function, particularly a time-varying weight.*

Another line of decision-making research that may help solve the LVO problem is the **non-stationary type of methods**. These methods ignore the internal structure of sub-objectives and treat the action-wise main objective as a non-stationary target. Thus, the LVO problem can be viewed as a special case of non-stationary multi-armed bandits, where each action corresponds to a non-stationary reward or loss. Auer et al. [5] propose the Exp3.S algorithm, which guarantees an optimal $O(\sqrt{S_T T})$ regret bound for adversarial bandits, where S_T is the number of distribution shifts by time T. Kocsis and Szepesvári [20] propose the D-UCB algorithm for stochastic non-stationary bandits, introducing a discount factor so that recent observations have more weight than older ones in the estimation. Garivier and Moulines [14] further prove that the D-UCB algorithm shares the same optimality as Exp3.S in the stochastic non-stationary bandits problem. Additionally, Garivier and Moulines [14] propose the SW-UCB algorithm, which considers only the most recent observations within a fixed-length sliding window for reward estimation, and prove that SW-UCB shares the same-order optimality as Exp3.S and D-UCB. For the contextual extension of LVO, the linear structure in (7) is very similar to the linear structural assumption in stochastic linear bandits [1, 2], especially non-stationary stochastic linear bandits [9] when the reward distribution changes over time. Inspired by the sliding window and discounted ideas of Garivier and Moulines [14] and Kocsis and Szepesvári [20], Cheung et al. [9] propose the SW-LinUCB algorithm, and Russac et al. [28] propose the D-LinUCB algorithm. Subsequently, Zhao et al. [41] propose the RestartUCB algorithm, which restarts the reward estimation whenever the distribution changes significantly. All the above algorithms guarantee a sub-optimal $O(P_T^{\frac{1}{4}} T^{\frac{3}{4}})$ regret, where P_T is the path length of the linearization parameter that quantifies the extent of distribution changes. Wei and Luo [36] improve the bound to an optimal $O(P_T^{\frac{1}{3}} T^{\frac{2}{3}})$ by combining a non-stationarity detector with base algorithms that perform well in stationary environments. The central difficulty lies in handling the arbitrarily changing weight vectors. In the worst case, the number of switching times can be $S_T = \Theta(T)$, resulting in an $O(T)$ regret bound that is linearly dependent on time T. This implies that the algorithm does not eventually converge to the correct action. A similar difficulty arises in the contextual case, where $P_T = \Theta(T)$ leads to a linear regret bound of $O(T)$. Therefore, directly applying existing algorithms for non-stationary bandits is infeasible in the worst case. However, in milder scenarios where S_T or P_T are smaller, there remains a possibility that this approach could partially address the LVO problem.