Multi-Objective Actor-Critics for Real-Time Bidding in Display Advertising*

Haolin Zhou[1]**, Chaoqi Yang[2]**, Xiaofeng Gao[1](✉), Qiong Chen[3], Gongshen Liu[4], and Guihai Chen[1]

[1] MoE Key Lab of Artificial Intelligence,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China
[2] University of Illinois at Urbana-Champaign, Illinois, USA
[3] Tencent Ads, Beijing, China
[4] School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University, Shanghai, China
{koziello, lgshen}@sjtu.edu.cn, chaoqiy2@illinois.edu
{gao-xf, gchen}@cs.sjtu.edu.cn, evechen@tencent.com

Abstract. Online Real-Time Bidding (RTB) is a complex auction game among which advertisers struggle to bid for ad impressions when a user request occurs. Considering display cost, Return on Investment (ROI), and other influential Key Performance Indicators (KPIs), large ad platforms try to balance the trade-off among various goals in dynamics. To address the challenge, we propose a **M**ulti-**O**bjec**Ti**ve **A**ctor-**C**ritics algorithm based on reinforcement learning (RL), named MoTiAC, for the problem of bidding optimization with various goals. In MoTiAC, objective-specific agents update the global network asynchronously with different goals and perspectives, leading to a robust bidding policy. Unlike previous RL models, the proposed MoTiAC can simultaneously fulfill multi-objective tasks in complicated bidding environments. In addition, we mathematically prove that our model will converge to Pareto optimality. Finally, experiments on a large-scale real-world commercial dataset from Tencent verify the effectiveness of MoTiAC versus a set of recent approaches.

Keywords: Real-time Bidding · Reinforcement Learning · Display Advertising · Multiple Objectives

1 Introduction

The rapid development of the Internet and smart devices has created a decent environment for the advertisement industry. As a result, real-time bidding (RTB) has gained

* This work was supported by the National Key R&D Program of China [2020YFB1707903]; the National Natural Science Foundation of China [61872238, 61972254]; Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102]; the CCF-Tencent Open Fund [RAGR20200105] and the Tencent Marketing Solution Rhino-Bird Focused Research Program [FR202001]. Chaoqi Yang completed this work when he was an undergraduate student at Shanghai Jiao Tong University. Xiaofeng Gao is the corresponding author.

** Equal contribution.

continuous attention in the past few decades [23]. A typical RTB setup consists of publishers, supply-side platforms (SSP), data management platforms (DMP), ad exchange (ADX), and demand-side platforms (DSP). When an online browsing activity triggers an ad request in one bidding round, the SSP sends this request to the DSP through the ADX, where eligible ads compete for the impression. The bidding agent, DSP, represents advertisers to come up with an optimal bid and transmits the bid back to the ADX (e.g., usually within less than 100ms [23]), where the winner is selected to be displayed and charged by a generalized second price (GSP).

In the RTB system, *bidding optimization* in DSP is regarded as the most critical problem [24]. Unlike Sponsored Search (SS) [25], where advertisers make keyword-level bidding decisions, DSP in the RTB setting needs to calculate the optimal impression-level bidding under the basis of user/customer data (e.g., income, occupation, purchase behavior, gender, etc.), target ad (e.g., content, click history, budget plan, etc.) and auction context (e.g., bidding history, time, etc.) in every single auction [24].

Thus, our work focuses on DSP, where *bidding optimization* happens. In real-time bidding, two fundamental challenges need to be addressed. Firstly, the RTB environment is highly dynamic. In [20,24,26], researchers make a strong assumption that the bidding process is stationary over time. However, the sequence of user queries (e.g., incurring impressions, clicks, or conversions) is time-dependent and mostly unpredictable [25], where the outcome influences the next auction round. Traditional algorithms usually learn an independent predictor and conduct fixed optimization that amounts to a greedy strategy, often not leading to the optimal return [3]. Agents with reinforcement learning (RL) address the aforementioned challenge to some extent [7,12,25]. RL-based methods can alleviate the instability by learning from immediate feedback and long-term reward. However, these methods are limited to either *Revenue* or *ROI*, which is only one part of the overall utility. In the problem of RTB, we assume that the utility is two-fold, as outlined: (i) the cumulative cost should be kept within the budget; (ii) the overall revenue should be maximized. Therefore, the second challenge is that the real-world RTB industry needs to consider multiple objectives, which are not adequately addressed in the existing literature.

To address the challenges mentioned above, we propose a *Multi-Objective Actor-Critic* model, named `MoTiAC`. We generalize the popular asynchronous advantage actor-critic (A3C) [13] reinforcement learning algorithm for multiple objectives in the RTB setting. Our model employs several local actor-critic networks with different objectives to interact with the same environment and then updates the global network asynchronously according to different reward signals. Instead of using a fixed linear combination of different objectives, `MoTiAC` can decide on adaptive weights over time according to how well the current situation conforms with the agent's prior. We evaluate our model on click data collected from the Tencent ad bidding system. The experimental results verify the effectiveness of our approach versus a set of baselines.

The contributions in this paper can be summarized as follows:

- We identify two critical challenges in RTB and are well motivated to use multi-objective RL as the solution.
- We propose a novel multi-objective actor-critic model `MoTiAC` for optimal bidding and prove the superiority of our model from the perspective of Pareto optimality.

– Extensive experiments on a real industrial dataset collected from the Tencent ad system show that `MoTiAC` achieves state-of-the-art performance.

2 Preliminaries

2.1 Definition of oCPA and Bidding Process

In the online advertising scenario, there are three main ways of pricing. Cost-per-mille (CPM) [7] is the first standard, where revenue is proportional to *impression*. Cost-per-click (CPC) [24] is a performance-based model, i.e., only when users *click* the ad can the platform get paid. In the cost-per-acquisition (CPA) model, the payment is attached to each *conversion* event. Regardless of the pricing ways, ad platforms always try to maximize revenue while simultaneously maintaining the overall cost within the budget predefined by advertisers.

In this work, we focus on one pricing model that is currently used in Tencent online ad bidding systems, called optimized cost-per-acquisition (oCPA), in which **advertisers are supposed to set a target CPA price, denoted by $\textbf{CPA}_{\textbf{target}}$ for each conversion while the charge is based on each click**. The critical point for the bidding system is to make an optimal strategy to allocate overall impressions among ads properly, such that (i) the real click-based cost is close to the estimated cost calculated from $\text{CPA}_{\text{target}}$, specifically,

$$\#\text{clicks} \times \text{CPC}_{\text{next}} \approx \#\text{conversions} \times \text{CPA}_{\text{target}}, \tag{1}$$

where CPC_{next} is the cost charged by the second highest price and $\text{CPA}_{\text{target}}$ is pre-defined for each conversion; (ii) more overall conversions. In the system, the goal of our bidding agent is to generate an optimal $\textbf{CPC}_{\textbf{bid}}$ price, adjusting the winner of each impression. We denote $\mathcal{I} = \{1, 2, ..., n\}$ as bidding iterations, $\mathcal{A} = \{ad_1, ad_2, ...\}$ as a set of all advertisements. For each iteration $i \in \mathcal{I}$, $ad_j \in \mathcal{A}$, our bidding agent will decide on a $\text{CPC}_{\text{bid}}^{(i,j)}$ to play auction. Then the ad with the highest $\text{CPC}_{\text{bid}}^{(i,j)}$ wins the impression and then receives possible $\#\text{clicks}^{(i,j)}$ (charged by $\text{CPC}_{\text{next}}^{(i,j)}$ per click) and $\#\text{conversions}^{(i,j)}$ based on user engagements.

2.2 Optimization Goals in Real-Time Bidding

On the one hand, when CPC_{bid} is set higher, ads are more likely to win this impression to get clicks or later conversions, and vice versa. However, on the other hand, higher CPC_{bid} means lower opportunities for other ad impressions. Therefore, to determine the appropriate bidding price, we define the two optimization objectives as follows:

Objective 1: Minimize overall CPA. The first objective in RTB bidding problem is to allocate impression-level bids in every auction round, so that each ad will get reasonble opportunities for display and later get clicks or conversions, which makes $\textbf{CPA}_{\textbf{real}}$ close to $\textbf{CPA}_{\textbf{target}}$ pre-defined by the advertisers:

$$\textbf{CPA}_{\textbf{real}}^{(j)} = \frac{\sum_{i\in\mathcal{I}} \#\text{clicks}^{(i,j)} \times \text{CPC}_{\text{next}}^{(i,j)}}{\sum_{i\in\mathcal{I}} \#\text{conversions}^{(i,j)}}, \quad \forall ad_j \in \mathcal{A}. \tag{2}$$

To achieve the goal of minimizing overall CPA, i.e., be in line with the original budget, a lower ratio between $\mathbf{CPA}^{(j)}_{\mathbf{real}}$ and $\mathbf{CPA}^{(j)}_{\mathbf{target}}$ is desired. Precisely, when the ratio is smaller than 1, the agent will receive a positive feedback. On the contrary, when the ratio is greater than 1, it means that the actual expenditure exceeds the budget and the agent will be punished by a negative reward.

Objective 2: Maximize conversions. The second objective is to enlarge conversions as much as possible under the condition of a reasonable CPA_{real}, so that platform can stay competitive and run a sustainable business:

$$\textbf{\#conversions}^{(j)} = \sum_{i \in \mathcal{I}} \text{\#conversions}^{(i,j)}, \quad \forall ad_j \in \mathcal{A}, \tag{3}$$

where $\textbf{\#conversions}^{(j)}$ is a cumulative value until the current bidding auction. Obviously, relatively high #conversions will receive a positive reward. When the policy network gives fewer conversions, the agent will be punished with a negative reward.

Note that in the real setting, optimization objectives used by advertising platforms can be adjusted based on actual business needs. In the implementation and evaluation of `MoTiAC`, we use **ROI** (Return on Investment) and **Revenue**, corresponding to the two objectives for optimization, i.e., minimizing overall CPA and maximizing the number of conversions. Their definition will be detailed in Sec. 4.1.

3 Methodology

As shown in Sec. 2.1, the RTB problem is a multi-objective optimization problem. We need to control advertisers' budgets and make profitable decisions for the ad platform. Traditional RTB control policy or RL agent can hardly handle these challenges. In this work, we design `MoTiAC` to decouple the training procedure of multiple objectives into disentangled worker groups of actor-critics. We will elaborate on the technical details of `MoTiAC` in the following subsections.

3.1 Asynchronous Advantage Actor-Critic Model in RTB

An actor-critic reinforcement learning setting [8] in our RTB scenario consists of:

– **state** s**:** each state is composed of anonymous feature embeddings extracted from the user profile and bidding environment, indicating the current bidding state.
– **action** a**:** action is defined as the bidding price for each ad based on the input state. Instead of using discrete action space [20], our model outputs a distribution so that action can be sampled based on probability.
– **reward** r**:** obviously, the reward is a feedback signal from the environment to evaluate how good the previous action is, which guides the RL agent towards a better policy. In our model, we design multiple rewards based on different optimization goals. Each actor-critic worker group deals with one type of reward from the environment and later achieves multiple objectives together.

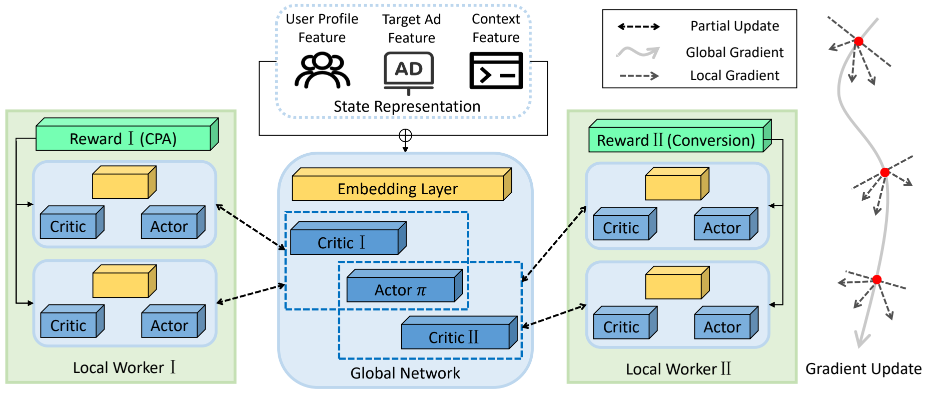

Fig. 1: Framework of the proposed `MoTiAC` in RTB.

– **policy** $\pi_\theta(\cdot)$**:** policy is represented as $\pi_\theta(a_t|s_t)$, which denotes the probability to take action a_t under state s_t. In an actor-critic thread, actor works as a policy network, and critic stands for value function $V(s;\theta_v)$ of each state. The parameters are updated according to the experience reward obtained during the training process.

For each policy π_θ, we define the utility function as

$$U(\pi_\theta) = E_{\tau\sim p_\theta(\tau)}[R(\tau)], \tag{4}$$

where $p_\theta(\tau)$ denotes the distribution of trajectories under policy π_θ, and $R(\tau)$ is a return function over trajectory τ, calculated by summing all the reward signals in the trajectory. The utility function is used to evaluate the quality of an action taken in a specific state. We also introduce value function from critic to reduce the varaition that may occur when updating parameters in real time. After collecting a number of tuples (s_t, a_t, r_t) from each trajectory τ, the policy network $\pi_\theta(\cdot)$ is updated by

$$\theta \leftarrow \theta + \eta_{actor} \sum_{t=1}^{T} (R(s_t) - V(s_t))\nabla_\theta \log \pi_\theta(a_t \mid s_t), \tag{5}$$

where η_{actor} represents the learning rate of policy network, T is a preset maximum step size in a trajectory, $R(s_t) = \sum_{n=t}^{T} \gamma^{n-t} r_n$ denotes the cumulative discounted reward, and γ is a decaying factor. The critic network, $V(s;\theta_v)$, could also be updated by:

$$\theta_v \to \theta_v + \eta_{critic} \frac{\partial \left(R_t - V_{\theta_v}(s_t)\right)^2}{\partial \theta_v}, \tag{6}$$

where η_{critic} represents the learning rate of value function.

3.2 Adaptive Reward Partition

In this subsection, we consider the general K-objective case, where K is the total number of objectives. As stated in Sec. 2.2, multiple objectives should be considered in modeling the RTB problem. One intuitive way [14] of handling multiple objectives is to integrate them into a single reward function linearly, and we call it *Reward Combination*: (i) A linear combination of rewards is firstly computed, where w_k quantifies the relative importance of the corresponding objective $R_k(\cdot)$:

$$R(s) = \sum_{k=1}^{K} w_k \times R_k(s). \tag{7}$$

(ii) A single-objective agent is then defined with the expected return equal to value function $V(s)$. However, a weighted combination is only valid when objectives do not compete [17]. In the RTB setting, the relationship between objectives can be complicated, and they usually conflict on different sides. The intuitive combination might flatten the gradient for each objective, and thus the agent is likely to limit itself within a narrow boundary of search space. Besides, a predefined combination may not be flexible in the dynamic bidding environment. Overall, such a *Reward Combination* method is unstable and inappropriate for the RTB problem, as we will show in the experiments.

Reward Partition. We now propose the *Reward Partition* scheme in `MoTiAC`. Specifically, we design reward for each objective and employ one group of actor-critic networks with the corresponding reward. There is one global network with an actor and multiple critics in our model. At the start of one iteration, each local network copies parameters from a global network. Afterward, local networks from each group will begin to explore based on their objective and apply weighted gradients to the actor and one of the critics (partial update) in the global network asynchronously, as shown in Fig. 1. Formally, we denote the total utility and value function of the k^{th} group ($k = 1, \cdots, K$) as $U^k(\pi_\theta)$ and $V_k(s; \theta_v)$, respectively. Different from the original Eqn. (5), the parameter updating of policy network in one actor-critic group of `MoTiAC` is formulated as

$$\theta \leftarrow \theta + \eta_{actor} w_k \sum_{t=1}^{T} (R_k(s_t) - V_k(s_t)) \nabla_\theta \log \pi_\theta(a_t \mid s_t), \tag{8}$$

where w_k is an objective-aware customized weight for optimization in range (0,1) and is tailored for each $ad_j \in A$. We can simply set w_k as

$$w_k = \frac{R_k(s_t) - V_k(s_t)}{\sum_{l=1}^{K} (R_l(s_t) - V_l(s_t))}, \tag{9}$$

while dynamically adjusting the value of w_k by giving higher learning weights to the local network that contributes more to the total reward. Motivated by Bayesian RL [5], we can generalize the customized weight and parameterize w_k by introducing a latent multinomial variable ϕ with $w_k = p(\phi = k|\tau)$ under trajectory τ, named as *agent's prior*. We set the initial prior as

$$p(\phi = k|\tau_0) = \frac{1}{K}, \quad \forall\, k = 1, 2, \ldots, K, \tag{10}$$

where τ_0 indicates that the trajectory just begins. When τ_t is up to state s_t, i.e., $\tau_t = \{s_1, a_1, r_1, s_2, a_2, r_2, \dots s_t\}$, we update the posterior by

$$p(\phi = k|\tau_t) = \frac{p(\tau_t|\phi = k)p(\phi = k)}{\sum_k p(\tau_t|\phi = k)p(\phi = k)}, \tag{11}$$

where $p(\tau_t|\phi = k)$ tells how well the current trajectory agrees with the utility of objective k. Based on priority factor w_k, together with the strategy of running different exploration policies in different groups of workers, the overall changes being made to the global actor parameters θ are likely to be less correlated and more objective-specific in time, which means our model can make wide exploration and achieve a balance between multiple objectives with a global overview.

In addition, we present some analysis for the two reward aggregation methods in terms of parameters update and value function approximation. If we attach the weights of *Reward Combination* to the gradients in *Reward Partition*, the parameters updating strategy should be identical on average. For *Reward Combination*, global shared actor parameters θ is updated by

$$\theta \leftarrow \theta + \eta_{actor} \sum_t \left(\left(\sum_{k=1}^{K} w_k \times R_k(s_t) - V_k(s_t) \right) \times \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right),$$

while in *Reward Partition*, the expected global gradient is given as

$$\theta \leftarrow \theta + \eta_{actor} \sum_t \left(\left(\sum_{k=1}^{K} (w_k \times R_k(s_t) - w_k \times V_k(s_t)) \right) \times \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right).$$

The difference between the two reward aggregating methods lies in the advantage part. Thus the effect of parameter updates heavily depends on how well and precisely the critic can learn from its reward. By learning in a decomposed manner, the proposed *Reward Partition* advances the *Reward Combination* by using easy-to-learn functions to approximate single rewards, thus yielding a better policy.

3.3 Optimzation and Training Procedure

In the framework of `MoTiAC`, the policy network explores continuous action space and outputs action distribution for inference. Therefore, loss for a single actor-critic worker (objective-k) is gathered from actor θ, critic θ_v, and action distribution entropy H to improve exploration by discouraging premature convergence to sub-optimal [13],

$$L_{\theta,\theta_v} = \eta_{actor} E[R(\tau)] + \eta_{critic} \sum_{s_t \in \tau} \|V_{\theta_v}(s_t) - R(s_t)\|^2 + \beta \sum_{s_t \in \tau} H(\pi(s_t)), \tag{12}$$

where β represents the strength of entropy regularization.

After one iteration (e.g., 10-minute bidding simulation), we compute gradients for each actor-critic network and push the weighted gradients to the global network. With multiple actor-learners applying online updates in parallel, the global network could explore to achieve a robust balance between multiple objectives. The training procedure of `MoTiAC` is shown in Algorithm 1.

Algorithm 1: Training for each actor-critic thread in MoTiAC

```
// Assume global shared parameters $\theta$ and $\theta_v$;
// Assume objective-specific parameters $\theta'_k$ and $\theta'_{v,k}$, $k \in \{1, 2, \ldots, K\}$;
Initialize step counter $t \leftarrow 1$; epoch $T$; discounted factor $\gamma$;
while $t < T_{max}$ do
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$ ;
    Synchronize specific parameters $\theta'_k = \theta$ and $\theta'_{v,k} = \theta_v$;
    Get state $s_t$ extracted from user profile features and bidding environment;
    // Assume ad set $\mathcal{A} = \{ad_1, ad_2, ...\}$;
    for $ad_j \in \mathcal{A}$ do
        repeat
            Determine bidding price $a_t$ according to policy $\pi(a_t \mid s_t; \theta'_k)$;
            Receive reward $r_t$ w.r.t objective k;
            Reach new state $s_{t+1}$;
            $t \leftarrow t + 1$;
        until terminal state;
        for $n \in \{t-1, ..., 1\}$ do
            $r_n \leftarrow r_n + \gamma \times r_{n+1}$;
            // Accumulative gradient w.r.t $\theta'_k$;
            $d\theta'_k \leftarrow d\theta'_k + \eta_{actor} \sum (r_n - V(s_n; \theta'_{v,k})) \nabla_{\theta'_k} \log \pi(a_n|s_n) +$
              $\beta \sum \nabla_{\theta'_k} H(\pi(a_n|s_n))$;
            // Accumulative gradient w.r.t $\theta'_{v,k}$;
            $d\theta'_{v,k} \leftarrow d\theta'_{v,k} + \eta_{critic} \sum \partial \|r_n - V(s_n; \theta'_{v,k})\|^2 / \partial \theta'_{v,k}$;
        end
        // Asynchronously update $\theta$ and $\theta_v$ with $d\theta'_k$ and $d\theta'_{v,k}$;
        // Compute $w_k = p(\phi = k|\tau)$ w.r.t objective $k$;
        $\theta \leftarrow \theta + w_k \times d\theta'_k$ and $\theta_v \leftarrow \theta_v + w_k \times d\theta'_{v,k}$;
    end
end
```

3.4 Convergence Analysis of **MoTiAC**

In this section, we use a toy demonstration to provide insights into the convergence property for the proposed MoTiAC. As illustrated in Fig. 2, the solid black line is the gradient contour of *objective 1*, and the black dash line is for *objective 2*. The yellow area within their intersection is the area of the optimal strategy, where both advertisers and publishers satisfy with their benefits. Due to the highly dynamic environment of RTB [3], the optimal bidding strategy will change dramatically.

Traditionally in a multi-objective setting, when people use linear combinations or other more complex transformations [11], like policy votes [18] of reward functions. They implicitly assume that the optimal solution is fixed, as shown in the upper part of Fig. 2. Consequently, their models can only learn the *initial optimal* and fail to characterize the dynamics. However, according to the dynamic environment in RTB, our MoTiAC adjusts the gradient w.r.t each possible situation towards a new optimal based on each objective separately and will easily be competent for real-world instability. Each gradient w.r.t the objectives forces the agent closer to the optimal for compensa-

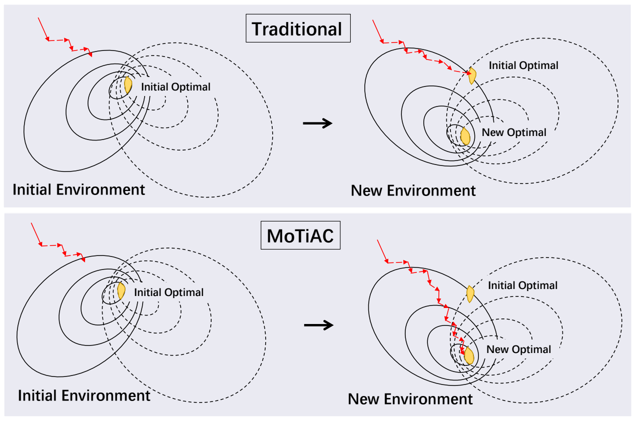

Fig. 2: Convergence illustration of `MoTiAC`.

tion rather than conflicts. Finally, the agent would reach the area of *new optimal* and tunes its position in the micro-level, called convergence.

We further prove that the global policy will converge to the Pareto optimality between these objectives. The utility expectation of the objective k is denoted as $E[U^k(\pi_\theta)]$. We begin the analysis with Theorem 1 [10],

Theorem 1. *(Pareto Optimality). If π^* is a Pareto optimal policy, then for any other policy π, one can at least find one k, so that $0 < k \leq K$ and,*

$$E[U^k(\pi^*)] \geq E[U^k(\pi)]. \tag{13}$$

The multi-objective setting assumes that the possible policy set $\varPi$ spans a convex space (K-simplices). The optimal policy of any affine interpolation of objective utility will be also optimal [4]. We restate in Theorem 2 by only considering the non-negative region.

Theorem 2. *π^* is Pareto optimal iff there exits $\{l_k > 0 : \sum_k l_k = 1\}$ such that,*

$$\pi^* \in \arg\max_\pi \left[\sum_k l_k E[U^k(\pi)]\right]. \tag{14}$$

Proof. We derive the gradient by aggregating Eqn. (8) as,

$$\nabla = \sum_{\tau_t}\sum_k p(\phi = k|\tau_t)\nabla_\theta U^k(\tau_t;\pi_\theta) \propto \sum_k p(\phi = k)\sum_{\tau_t} p(\tau_t|\phi = k)\nabla_\theta U^k(\tau_t;\pi_\theta)$$
$$= \sum_k p(\phi = k)\nabla_\theta E_{\tau_t}[U^k(\tau_t;\pi_\theta)] = \nabla_\theta\left[\sum_k p(\phi = k)E_{\tau_t}[U^k(\tau_t;\pi_\theta)]\right]. \tag{15}$$

By making $l_k = p(\phi = k)$ (Note that $\sum_k p(\phi = k) = 1$), we find that the overall gradient conform with the definition of Pareto optimality in Eqn. (14). Therefore, we conclude that `MoTiAC` converges to Pareto optimal, indicating that it can naturally balance different objectives.

Date	# of Ads	# of clicks	# of conversions
20190107	10,201	176,523,089	3,886,155
20190108	10,416	165,676,734	3,661,060
20190109	10,251	178,150,666	3,656,714
20190110	9,445	157,084,102	3,287,254
20190111	10,035	181,868,321	3,768,247

Table 1: Statistics of click data from Tencent bidding system.

4 Experiments

In the experiment, we use real-world industrial data to answer the following three research questions:

- **RQ1:** How does `MoTiAC` perform compared with other baseline methods?
- **RQ2:** What is the best way to aggregate multiple objectives?
- **RQ3:** How does `MoTiAC` balance the exploration of different objectives?

4.1 Experiment Setup

Dataset. In the experiment, the dataset is collected from the real-time commercial ads bidding system of Tencent. There are nearly 10,000 ads daily with a huge volume of click and conversion logs. According to real-world business, the bidding interval is set to be 10 minutes (144 bidding sessions for a day), which is much shorter than one hour [7]. Basic statistics can be found in Table 1.
Compared Baselines. We carefully select related methods for comparison and adopt the same settings for all the compared methods with 200 iterations. Details about implementation can be seen in Appendix A.2.

- **Proportional-Integral-Derivative (PID):** [2] is a widely used feedback control policy, which produces the control signal from a linear combination of proportional, integral, and derivative factors.
- **Advantage Actor-Critic (A2C):** [13] makes the training process more stable by introducing an advantage function. [7] generalizes the actor-critic structure in the RTB setting.
- **Deep Q-Network (DQN):** [20] uses DQN with a single objective under the assumption of consistent state transition in the RTB problem, while the similar structure can also be coupled with a dynamic programming approrach [3].
- **Aggregated A3C (Agg-A3C):** Agg-A3C [13] is proposed to disrupt the correlation of training data by introducing an asynchronous update mechanism.

We linearly combine multiple rewards (following *Reward Combination*) for all the baselines. Besides, we adopt two variants of our model: *Objective1-A3C (O1-A3C)* and *Objective2-A3C (O2-A3C)*, by only considering one of the objectives. We use four days

Model	Relative Cost	Relative ROI	Relative Revenue	R-score
PID	1.0000	1.0000	1.0000	1.0000
A2C	1.0366 (+3.66%)	0.9665 (-3.35%)	1.0019 (+0.19%)	0.9742
DQN	0.9765 (-2.35%)	1.0076 (+0.76%)	0.9840 (-1.60%)	0.9966
Agg-A3C	1.0952 (+9.52%)	0.9802 (-1.98%)	1.0625 (+6.25%)	0.9929
O1-A3C	0.9580 (-4.20%)	1.0170 (+1.70%)	0.9744 (-2.56%)	1.0070
O2-A3C	1.0891 (+8.91%)	0.9774 (-2.26%)	1.0645 (+6.45%)	0.9893
`MoTiAC`	1.0150 (+1.50%)	1.0267 (+2.67%)	1.0421 (+4.21%)	1.0203

Table 2: Comparative results based on PID.

of data for training and another day for testing and then use the cross-validation strategy on the training set for hyper-parameter selection. Similar settings can be found in literature [20,26].
Evaluation Metrics. We clarify the objectives of our problem based on the collected data. In Sec. 2.2, we claim that our two objectives are: (1) *minimize overall CPA*; (2) *maximize conversions*. We refer to the industrial convention and redefine our goals in the experiments. *Revenue* is a common indicator for platform earnings, which turns out to be proportional to conversions. *Cost* is the money paid by advertisers, which also appears to be a widely accepted factor in online advertising. Therefore, without loss of generality, we reclaim our two objectives to be:

$$\text{Revenue}^{(j)} = \text{conversions}\ ^{(j)} \times \text{CPA}^{(j)}_{\text{target}}, \quad \text{Cost}^{(j)} = \#\text{clicks}^{(j)} \times \text{CPC}^{(j)}_{\text{next}}, \tag{16}$$

$$\max \mathbf{ROI} \leftarrow \max \sum_{Ad_j \in A} \frac{\text{Revenue}^{(j)}}{\text{Cost}^{(j)}}, \tag{17}$$

which corresponds to the first objective: *CPA goal*, and

$$\max \mathbf{Revenue} \leftarrow \max \sum_{Ad_j \in A} \text{Revenue}^{(j)}, \tag{18}$$

related to the second objective: *Conversion goal*.

For the two variants of `MoTiAC`, O1-A3C corresponds to maximizing ROI, while O2-A3C is related to maximizing Revenue. In addition to directly comparing these two metrics, we also use *R-score* proposed in [12] to evaluate the model performance. The higher the *R-score*, the more satisfactory the advertisers and platform will be. In the real-world online ad system, PID is currently used to control bidding. We employ it as a standard baseline, and most of the comparison results will be based on PID, i.e., $value \rightarrow \frac{value}{value_{PID}}$, except for Sec. 4.4.

4.2 RQ1: Comparison with Recent Baselines

We perform the comparison of `MoTiAC` with other approaches. The results are shown in Table 2. The values in the parentheses represent the percentage of improvement or

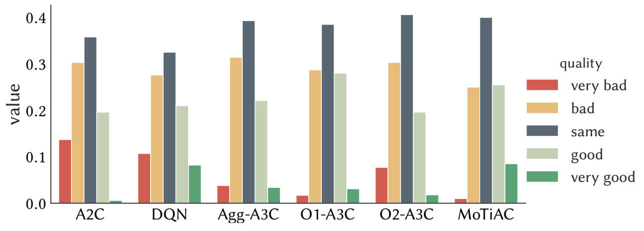

Fig. 3: Bidding quality distribution of compared methods over PID.

reduction towards PID. An optimal method is expected to improve both metrics (ROI & Revenue) compared with the current PID baseline.

Objective Comparison. We find that `MoTiAC` best balances the trade-off between two objectives (ROI & Revenue) based on the above considerations. Also, it has the highest R-score. Specifically, A2C is the worst since it gains a similar revenue (conversion goal) but a much lower ROI (CPA goal) than PID. The result proves that the A2C structure cannot fully capture the dynamics in the RTB environment. Based on a hybrid reward, DQN has a similar performance as O1-A3C, with relatively fewer conversions than other methods. We suspect the discrete action space may limit the policy to a local and unstable optimal. By solely applying the weighted sum in a standard A3C (Agg-A3C), the poor result towards ROI is not surprising. As the RTB environment varies continuously, fixing the formula of reward aggregation cannot capture the dynamic changes. It should be pointed out that two ablation models, O1-A3C and O2-A3C, present two extreme situations. O1-A3C performs well in the first ROI objective but performs poorly for the Revenue goal and vice versa for O2-A3C. By shifting the priority of different objectives over time, our proposed `MoTiAC` uses the agent's prior as a reference to make the decision in the future, precisely capturing the dynamics of the RTB sequence. Therefore, it outperforms all the other baselines.

Comparing *Reward Partition* and *Reward Combination*, the advantages of `MoTiAC` over other baselines show that our proposed method of accumulating rewards overall reduces the difficulty of agent learning and makes it easier for the policy network to converge around the optimal value.

Bidding Quality Analysis. To further verify the superiority of `MoTiAC` compared to other methods, we analyze the relative bidding quality of these methods over PID. We group all the ads into five categories based on their bidding results. The detailed evaluation metrics can be found in Appendix A.1. As shown in Fig. 3, both A2C and O2-A3C present more bad results compared than good ones, indicating that these two models could not provide a gain for the existing bidding system at a finer granularity. O1-A3C has a relatively similar performance as PID, as they both aims at minimizing real *CPA*. We also find that DQN tends to make bidding towards either very good or very bad, once again demonstrating the instability of the method. Agg-A3C shares the same distribution pattern with O1-A3C and vanilla PID, which indicates that the

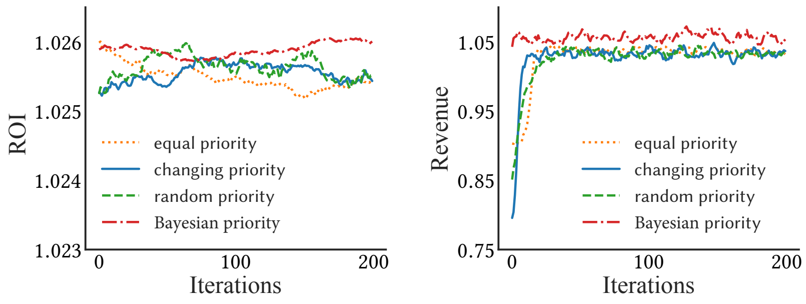

Fig. 4: Results under different priority functions.

combined reward does not work in our scenario. The proposed `MoTiAC` turns out to have a desirable improvement over PID with more ads on the right *good* side and fewer ads on the left *bad* side. Note that the negative transfer of multi-objective tasks makes some bidding results inevitably worse. However, we can still consider that `MoTiAC` can achieve the best balance among all the compared methods.

4.3 RQ2: Variants of w_k

To give a comprehensive view of `MoTiAC`, we perform different ways to aggregate objectives. Four different variants of w_k are considered in the experiment. Since we have two objectives, we use $w_1(t)$ for the first objective and $1 - w_1(t)$ for the second:

- equal priority: $w_1(t) = \frac{1}{2}$;
- changing priority: $w_1(t) = \exp(-\alpha \cdot t)$ with a scalar α;
- random priority: $w_1(t) = \text{random}([0, 1])$;
- Bayesian priority: One can refer to Eqn. (11).

As shown in Fig. 4, we present the training curves for ROI and Revenue. The first three strategies are designed before training and will not adjust to the changing environment. It turns out that they perform similarly in both objectives and could gain a decent improvement over the PID case by around +2.5% in ROI and +3% in Revenue. However, in *equal priority*, the curve of ROI generally drops when the iteration goes up, which stems from the fact that fixed equal weights cannot fit the dynamic environment. For *changing priority*, it is interesting that ROI first increases then decreases for priority shifting, as different priority leads to different optimal. In *random priority*, curves dramatically change in a small range since the priority function outputs the weight randomly. The *Bayesian priority* case, on the contrary, sets priority based on the conformity of the agent's prior and current state. Reward partition with agent prior dominates the first three strategies by an increasingly higher ROI achievement by +2.7% and better Revenue by around +4.2%.

4.4 RQ3: Case Study

In this section, we try to investigate how `MoTiAC` balances the exploration of multiple objectives and achieves the optimal globally. We choose one typical ad with large con-

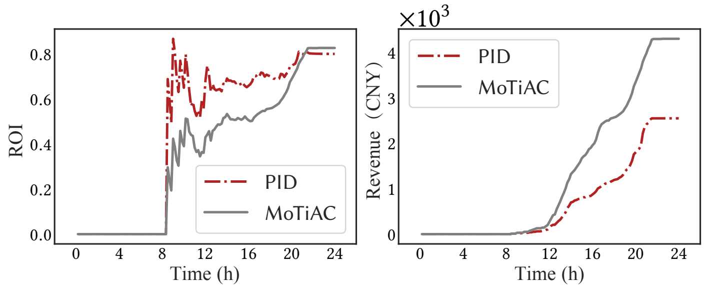

Fig. 5: ROI and Revenue curves of the target ad's reponse.

Models	Revenue (CNY)	Cost (CNY)	ROI
PID	3.184×10^3	2.548×10^3	0.8003
`MoTiAC`	4.298×10^3	5.199×10^3	0.8267

Table 3: Numerical results of the target ad using PID and `MoTiAC`.

versions and show the bidding process within 24 hours. As PID is the current model in the real ad system, we use PID to compare with `MoTiAC` and draw the results of ROI and Revenue curve in Fig. 5. We also collect the final numerical results in Table 3.

Fig. 5 shows a pretty low ROI initially. For the target ad, both models first try to lift the ROI. Based on the figure presented on the left, the red dashed curve rises from 0 to about 0.7 sharply for PID at 8h. The potential process should be that PID has given up most of the bid chances and only concentrates on those with a high conversion rate (CVR) so that we have witnessed a low Revenue gain of the PID model in the right figure from 8h to around 21h. Though the ROI curve remains relatively low, our `MoTiAC` can select good impression-level chances while considering the other objective. At 24h, `MoTiAC` finally surpasses PID in ROI because of the high volume of pre-gained Revenue. With long-term consideration, `MoTiAC` beats PID on both the cumulative ROI and Revenue. We can conclude that PID is greedy out of the immediate feedback mechanism. It is always concerned with the current situation and never considers further benefits. When the current state is under control, PID will appear conservative and give a shortsighted strategy, resulting in a seemingly good ROI and poor Revenue (like the red curve in Fig. 5). However, `MoTiAC` has a better overall view. It foresees the long-run benefit and will keep exploration even temporarily deviating from the right direction or slowing down the rising pace (ROI curve for the target ad at 8h). Under a global overview, `MoTiAC` can finally reach better ROI and Revenue than PID.

5 Related Work

Real-time Bidding. Researchers have proposed static methods [15] for optimal biddings, such as constraint optimization [24], to perform an impression-level evaluation.

However, traditional methods inevitably ignore that real-world situations in RTB are often dynamic [21] due to the unpredictability of user behavior [7] and different marketing plans [22] from advertisers. Furthermore, the auction process of optimal bidding is formulated as a Markov decision process (MDP) in recent study [7,12]. Considering the various goals of different players in RTB, a robust framework is required to balance these multiple objectives. Therefore, we are motivated to propose a novel multi-objective RL model to maximize the overall utility of RTB.

Reinforcement Learning. Significant achievements have been made by the emergence of RL algorithms, such as policy gradient [19] and actor-critic [8]. With the advancement of GPU and deep learning (DL), more successfully deep RL algorithms [9,13] have been proposed and applied to various domains. Meanwhile, there are previous attempts to address the multi-objective reinforcement learning (MORL) problem [6], where the objectives are combined mainly by static or adaptive linear weights [1,14] or captured by a set of policies and evolving preferences [16].

6 Conclusion and Future Directions

In this paper, we propose **M**ulti-**O**bjec**Ti**ve **A**ctor-**C**ritics for real-time bidding in display advertising. `MoTiAC` utilizes objective-aware actor-critics to solve the problem of multi-objective bidding optimization. Our model can follow adaptive strategies in a dynamic RTB environment and outputs the optimal bidding policy by learning priors from historical data. We conduct extensive experiments on the real-world industrial dataset. Empirical results show that `MoTiAC` achieves state-of-the-art on the Tencent advertising dataset. One future direction could be extending multi-objective solutions with priors in the multi-agent reinforcement learning area.

References

1. A., A., M., D., L., T., N., A., S., D.: Dynamic weights in multi-objective deep reinforcement learning. In: International Conference on Machine Learning (ICML). pp. 11–20 (2019)
2. Bennett, S.: Development of the pid controller. IEEE control systems pp. 58–62 (1993)
3. Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., Guo, D.: Real-time bidding by reinforcement learning in display advertising. In: Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM). pp. 661–670 (2017)
4. Critch, A.: Toward negotiable reinforcement learning: shifting priorities in pareto optimal sequential decision-making. arXiv:1701.01302 (2017)
5. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al.: Bayesian reinforcement learning: A survey. Foundations and Trends® in Machine Learning **8**(5-6), 359–483 (2015)
6. Hayes, C.F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L.M., Dazeley, R., Heintz, F., et al.: A practical guide to multi-objective reinforcement learning and planning. arXiv preprint arXiv:2103.09568 (2021)
7. Jin, J., Song, C., Li, H., Gai, K., Wang, J., Zhang, W.: Real-time bidding with multi-agent reinforcement learning in display advertising. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM) (2018)
8. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1008–1014 (2000)

9. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015)
10. Lin, X., Zhen, H.L., Li, Z., Zhang, Q.F., Kwong, S.: Pareto multi-task learning. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 12060–12070 (2019)
11. Lizotte, D.J., Bowling, M.H., Murphy, S.A.: Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In: International Conference on Machine Learning (ICML). pp. 695–702 (2010)
12. Lu, J., Yang, C., Gao, X., Wang, L., Li, C., Chen, G.: Reinforcement learning with sequential information clustering in real-time bidding. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM). pp. 1633–1641 (2019)
13. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning (ICML). pp. 1928–1937 (2016)
14. Pasunuru, R., Bansal, M.: Multi-reward reinforced summarization with saliency and entailment. arXiv:1804.06451 (2018)
15. Perlich, C., Dalessandro, B., Hook, R., Stitelman, O., Raeder, T., Provost, F.: Bid optimizing and inventory scoring in targeted online advertising. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 804–812 (2012)
16. Pirotta, M., Parisi, S., Restelli, M.: Multi-objective reinforcement learning with continuous pareto frontier approximation. In: 29th AAAI Conference on Artificial Intelligence (AAAI) (2015)
17. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 525–536 (2018)
18. Shelton, C.R.: Balancing multiple sources of reward in reinforcement learning. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1082–1088 (2001)
19. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1057–1063 (2000)
20. Wang, Y., Liu, J., Liu, Y., Hao, J., He, Y., Hu, J., Yan, W.P., Li, M.: Ladder: A human-level bidding agent for large-scale real-time online auctions. arXiv:1708.05565 (2017)
21. Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X., Xu, J., Gai, K.: Budget constrained bidding by model-free reinforcement learning in display advertising. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM) (2018)
22. Xu, J., Lee, K.c., Li, W., Qi, H., Lu, Q.: Smart pacing for effective online ad campaign optimization. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 2217–2226 (2015)
23. Yuan, S., Wang, J., Zhao, X.: Real-time bidding for online advertising: measurement and analysis. In: Proceedings of the 7th International Workshop on Data Mining for Online Advertising (ADKDD). p. 3 (2013)
24. Zhang, W., Yuan, S., Wang, J.: Optimal real-time bidding for display advertising. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 1077–1086 (2014)
25. Zhao, J., Qiu, G., Guan, Z., Zhao, W., He, X.: Deep reinforcement learning for sponsored search real-time bidding. arXiv preprint arXiv:1803.00259 (2018)
26. Zhu, H., Jin, J., Tan, C., Pan, F., Zeng, Y., Li, H., Gai, K.: Optimized cost per click in taobao display advertising. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 2191–2200 (2017)

A Supplementary Material

A.1 Pricing Evaluation Metrics

We present our pricing evaluation metric below: Algorithm 2 is the pricing evaluation strategy based on our business, and in Algorithm 3 we specify the *Score* function.

Algorithm 2: Pricing Evaluation over PID.

```
// We use $Value_{method}$ to denote the value given by other approaches, and $Value_{PID}$
  denotes the value given by PID;
// Pricing score is given per ad;
for $ad_j \in A$ do
    // $CPA$ calculation;
    $\Delta cost = cost_{method} - cost_{PID}$;
    $\Delta conversion = conversion_{method} - conversion_{PID}$;
    $\Delta CPA = (\Delta cost + 0.5 \times CPA_{target}) \;/\; (\Delta conversion + 0.5)$;
    $CPA_{PID} = cost_{PID} \;/\; (conversion_{PID} + 0.01)$;
    $CPA_{method} = cost_{method} \;/\; (conversion_{method} + 0.01)$;
    // Bias calculation;
    $\Delta bias = \Delta CPA \;/\; CPA_{target} - 1$;
    $bias_{PID} = CPA_{PID} \;/\; CPA_{target} - 1$;
    $bias_{method} = CPA_{method} \;/\; CPA_{target} - 1$;
    // Enter the loop;
    if $bias_{PID} \leq 0.2$ then
        $\Delta loss = \Delta cost \;/\; cost_{PID}$;
        $score = Score(\Delta loss)$;
    else
        if $bias_{method} \leq 0.2 \bigwedge bias_{PID} \geq 0.3$ then
            $score = 2$;
        end
        $\Delta loss = \Delta cost \;/\; cost_{PID}$;
        if $\Delta bias \geq 0.3$ then
            $score = -Score(\Delta loss)$;
        else
            $score = Score(\Delta loss)$;
        end
    end
    Return score for $ad_j$;
end
```

A.2 Implementation Details

We shall present our implementation details for all baselines and our `MoTiAC` as follows:

Algorithm 3: Score Function

```
Input: $\Delta loss$
Output: $score$
function Score ($\Delta loss$) :
    if $\Delta loss \leq 0.1$ then
        Return 0 // similar performance
    else if $\Delta loss \leq 0.2$ then
        Return 1 // good / bad performance
    else
        Return 2 // very good / bad performance
    end
    // sign will be given outside
end
```

- **PID**: control signal is given by difference across two intervals in our experiment, in order to making sure less fluctuation and stable result. Ratio for proportional factor is 0.04 , for the integral factor is 0.25 and for the derivative factor is 0.002.
- **A2C, Agg-A3C**: in A2C, we use two hidden layers with 256 and 128 dimensions. Output for *Actor* is one-dimension scalar μ and σ (Gaussian Distribution), and then actions will be sampled from the above-mentioned distribution with boundaries. Output for *Critic* is one-dimension scalar. *State* is a normalized 17-dimension feature vector. Agg-A3C has the same architecture, but is on extension of A2C with 5 workers running in parallel and updates global net asynchronously.
- **DQN**: all settings in DQN are the same as in A2C and Agg-A3C, except for *Actor*. The output of *Actor* has 100 dimension w.r.t 100 values ranging from 0 to 1, and 0.01 as its interval. Then based on values we choose the best $r \in [0, 1]$, and gives $r \times$ CPA$_{target}$ as CPC$_{bid}$.
- **O1-A3C, O2-A3C**: all settings in O1-A3C and O2-A3C are the same as in A2C and Agg-A3C, except for reward. *Reward* in O1-A3C and O2-A3C is set according to their objectives.
- **`MoTiAC`**: we use two hidden layers with 256 and 128 dimensions for all nets in `MoTiAC`. However, the difference is that we have two critics in global nets with the same structures. And for each update period, one group of workers can only push their gradients to actor and one of the critics. We set 5 worker as an objective group. Also, `MoTiAC` is implemented with multiprocessing and shared memory.