Learning to Bid Long-Term: Multi-Agent Reinforcement Learning with Long-Term and Sparse Reward in Repeated Auction Games

J.Tan,[1] **R.Khalili,** [1] **H.Karl**[2]

[1]Huawei Technologies Munich Research Center, Munich Germany, jingtan@huawei.com
[2]Hasso Plattner Institute, Potsdam Germany, holger.karl@hpi.de

Abstract

We propose a multi-agent distributed reinforcement learning algorithm that balances between potentially conflicting short-term reward and sparse, delayed long-term reward, and learns with partial information in a dynamic environment. We compare different long-term rewards to incentivize the algorithm to maximize individual payoff and overall social welfare. We test the algorithm in two simulated auction games, and demonstrate that 1) our algorithm outperforms two benchmark algorithms in a direct competition, with cost to social welfare, and 2) our algorithm's aggressive competitive behavior can be guided with the long-term reward signal to increase both individual payoff and overall social welfare.

1 Introduction

Auction is a common resource allocation mechanism in e.g. networking (Xu et al. 2012b,a), energy (Lucas et al. 2013), e-commerce (Huang and Kauffman 2011), for its efficient price discovery in a dynamic market with partial information (Schindler et al. 2011; Einav et al. 2018). In many such applications using auction mechanism, agents are designed to represent the bidders and automatically bid in the auctions. The agents have private goals and valuations; they behave autonomously to maximize payoff. They also learn to continuously improve their strategy in a dynamic environment, based on other agents' strategies (Busoniu et al. 2008).

Reinforcement learning (RL) algorithms are often used in such applications, for their ability to learn with sparse environment feedback and balance between exploitation and exploration (Teng et al. 2013; Almasri et al. 2020). However, challenges remain. Firstly, although RL algorithms are often used to learn sequential tasks, many of them are still relatively "short-term": learning is based on a reward given immediately after action and state transition, and as the prediction horizon extends farther into the future, influence of the current action decreases exponentially. Moreover, if the reward is sparse and delayed, the reward estimation often has a high variance due to lack of predictable future states, especially with a big state-action space and variance in state value (Mataric 1994; Shahriari 2017). If decisions have long-term effects that are only apparent after a variable delay, or if short-term rewards conflict with long-term goals, such "short-term" algorithms would lead to worse performance in the long run. Secondly, many RL algorithms are designed for single agents, whereas the dynamic nature of a multi-agent environment requires tradeoff between optimality and convergence while keeping computation and communication complexity tractable (Feigenbaum et al. 2007). Also, with multiple agents, decentralized learning may lead to conflicts between social welfare (i.e. total reward of all agents) and individual gains (Almasri et al. 2020), this is especially undesirable in applications where public goods are distributed among agents, e.g., network resource and energy. We need a flexible way to incentivize agents to incorporate common goals, without using hard-coded behavior rules.

To address these challenges, we propose DRACO2, a multi-agent, long-term learning algorithm with credit assignment. We define a reward mechanism that decouples short-term dense and long-term sparse rewards and enables learning on different time scales. We use credit assignment to break down the long-term reward into a weight vector that is aligned with short-term rewards. In a dynamic and competitive environment, our core RL algorithm learns the best-response strategy updated in a fictitious self-play (FSP) method to improve convergence. Our learning agents have a state and reward-predictive model to increase prediction accuracy of the future (i.e. consequence of their actions). Finally, we use the curiosity-learning concept (Pathak et al. 2017), which has an adversarial setting to encourage the RL model to explore state-action space where the agent lacks predictive power.

To demonstrate the performance of DRACO2, we simulate two repeated auction games with learning-capable agents as bidders, and one single passive agent as a broker. The game setup is suitable for analyzing our algorithm, for it 1) creates a dynamic and competitive environment with independent agents, each with private values and goal; 2) has a vast state-action space; 3) creates conflicting short-term and long-term objectives: in the short term, the bidder is incentivized to receive immediate payoff, but in the long term, winning a bid would reduce the money available for future bids and bind the agent's resources, thus incurring opportunity cost; and 4) provides choices of long-term reward as incentive to bidders.

Empirical results show that DRACO2 outperforms both

the short-term algorithm and the vanilla curiosity learning algorithm in a direct competition, although at the cost of reduced social welfare. Then, to show the algorithm's flexibility in reacting to social welfare incentives, we use a fairness index score as external long-term reward to replace the original profit-seeking goal. As a result, all agents with DRACO2 receive maximum cumulated payoff – it is proof that agents can be motivated to achieve common goals without compromising their individual gains.

Our contributions:

- DRACO2 is extremely aggressive and competitive in both simulated auction games, outperforming benchmark algorithms, showing its capability to learn with sparse, delayed, sporadic reward and partial information in a dynamic, adversarial environment.
- Despite its aggressive behavior, it is easy to influence DRACO2 by simply replacing the profit-seeking goal with a fairness goal, compromising neither individual gain nor privacy.
- We open source our code (source 2021).

2 Related Work

One of the biggest challenges of applying RL in the real world is to learn behavior towards long-term goals with delayed and sparse reward signal (Dulac-Arnold et al. 2021). One common approach is to extract features from historical records, thus linking the delayed reward to behaviors in the past (Hester and Stone 2013). Learning with such algorithms is inefficient since learning from past experiences can only happen when the delayed outcomes become available. To address the delay, (A et al. 2018) factorizes one state into an intermediate and a final state with independent transition probabilities and predicts each state at different intervals. Reference (Hung et al. 2019) describes a credit-assignment method that focuses on the most relevant memory records via content-based attention; the algorithm is capable of locating past memory to execute new tasks and generalizes very well. These approaches focus more on the delay in reward signal and less on sparsity. In our setup, the long-term reward is delayed, sparse and sporadic.

To address sparsity of rewards, many model-based methods add intrinsic, intermediate rewards between sparse extrinsic reward signals. Such methods often adopt a supervised learning algorithm to predict next states and use the difference between the predicted and target state-action pair values as intrinsic reward. Although they propagate prediction inaccuracy into the future, they learn faster. For example, (Hester and Stone 2013) separately trains many "feature models" to predict each feature of the next state as well as a "reward model" to predict reward. Between sparse extrinsic rewards, the algorithm samples estimated next state and reward from the models. The models are only updated when there is new input available. Their approach assumes that state features are independent and can be learned separately, and the accuracy of the reward model is still related to the sparsity of the reward signal. (Burda et al. 2019) uses a long-short-term memory (LSTM) to extract features from past memory that are more relevant to the current task, thus improving the model's generalization properties. The algorithm also uses two independent models to predict next state and action, the prediction accuracy becomes intermediate, intrinsic rewards inserted between sparse extrinsic rewards. In this approach, the intrinsic reward signal is not related to the extrinsic sparse reward and the final outcome of the game is not credited to each of the agent's behaviors. The lack of credit assignment may affect learning efficiency, especially when there is conflict between the agent's short-term and long-term goals, as is the case in our setup.

Among learning algorithms for distributed decision making, no-regret algorithms apply to a wide range of problems and converge fast; however, they require the knowledge of best strategies that are typically assumed to be static (Chang 2007). Best-response algorithms search for best responses to other users' strategies, not for an equilibrium – they therefore adapt to a dynamic environment, but they may not converge at all (Weinberg et al. 2004). To improve the convergence property of best-response algorithms, (Bowling et al. 2002) introduces an algorithm with varying learning rate depending on the reward; (Weinberg et al. 2004) extends the work to non-stationary environments. However, both these algorithms provably converge only with restricted classes of games, and they are hard to implement in large or continuous state-action space, as is also the case in our set up of a multi-agent dynamic environment. The FSP method, on the other hand, addresses strategic agents' adaptiveness in a dynamic environment by incrementally evaluating state information and by keeping a weighted historical record (Heinrich et al. 2015), and it is easy to implement in a large state space. It therefore befits our requirements.

Table 1: Sec.3 symbol definition

Sym	Description	Sym	Description
$m \in M$	bidder	v	bid value
α	backoff decision	b	bidding price
c	joining cost	q	backoff cost
p	payment	z	bidding outcome
u	imediate payoff	U	cumulated payoff

3 Problem Formulation

We formulate the long-term reward maximization problem in a generalized repeated auction that can be first- or second-price, forward or reverse, with any customized winning rules and payment scheme. Table 1 summarizes the notation.

Let M be the set of bidders. Bidder $m \in M$ has at most 1 demand for the commodity at t, denoted as $m^t \in \{0,1\}$. Bidder m has two actions: whether to back off $\alpha_m^t \in \{0,1\}$, and which price to bid $b_m^t \in \mathbb{R}_+$; the bidder draws them from a strategy. Bidder m determines its bidding price b_m^t using some function f_m of its private valuation $v_m \in \mathbb{R}_+$ of the commodity, hence $b_m^t = f_m(v_m)$; f_m is only known to the bidder. The competing bidders draw their actions from a joint distribution π_{-m}^t based on $(p^1, \cdots, p^{t-1})$, where $p^t \in \mathbb{R}_+$ is the final price of the commodity at time t. The final price is a function of all bidding prices at time

Table 2: Sec.4.1 symbol definition

Sym	Description	Sym	Description
ζ	best response	ψ	behavioral strategy
$\mathbf{e}_m$	env. variables	ρ	private bidder info
$\mathbf{a}$	action, $\mathbf{a} = (\alpha, b)$	P^t_{-m}	other bidders state
sl^t_m	SL present state	rl^t_m	RL present state
S^t_m	RL complete state	$\bar{u}$	$\bar{u}$'s weight factor
θ	actor parameters	$\mathbf{w}$	critic parameters
γ	learning rate	δ	TD error
η	ζ's weight	ν	history length
μ	action mean	Σ	action covariance

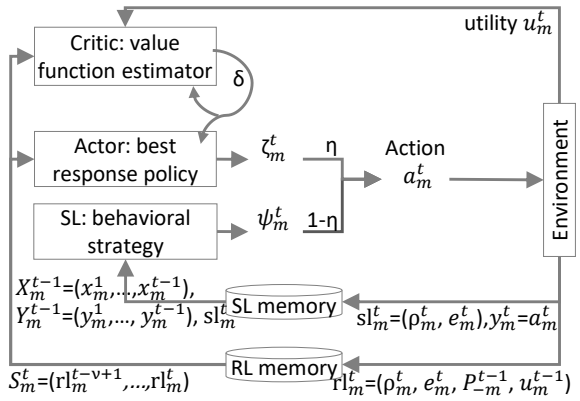


Figure 1: Short-term algorithm

t: $p^t = g(\mathbf{b}^t), \mathbf{b}^t \in \mathbb{R}_+^{|M|}$, depending on the auction mechanism we use; for example, in a first price lowest-bid auction, $g(\cdot) = \min(\cdot)$. Bidder m observes the new p^t as feedback. The commodity is granted to the bidder with the highest score according to the broker's internal logic, for example, in any lowest-bid auction, bidder m's score is a linearly decreasing function of b^t_m. m's utility is denoted by $u_m(b^t_m, z^t_m), z^t_m \in \{0,1\}$, if $z^t_m = 1$, m wins. A winning bidder receives an immediate payoff that is a function of m's private value v^t_m of the commodity, its bidding price b^t_m, and the final price p^t; losing bidders receive a negative payoff c^t_m as cost to join the auction, and bidders that backed off receive a negative payoff q^t_m as cost of backoff. We write the payoff as $u^t_m = h(v^t_m, b^t_m, z^t_m, p^t, c^t_m, q^t_m)$. The auction repeats for T periods. Each bidder's goal is to independently maximize its long-term utility: $\mathcal{U} = \frac{1}{T}\sum_{t=1}^{T} u_m(b^t_m), T \to \infty$.

4 Proposed Solution

To solve the long-term reward maximization problem described in Sec. 3, we propose an RL algorithm for long-term reward maximization. We first introduce the benchmark short-term algorithm in Section 4.1; that algorithm maximizes a short-term reward. Then, in Section 4.2, we introduce our long-term reward maximization algorithm that is based on the short-term algorithm.

Algorithm 1: FSP algorithm for bidder m

1: Initialize ψ_m, ζ_m arbitrarily, $t = 1, \eta = 1/t, \nu, P^{t-1}_{-m} = 0, u^{t-1}_m = 0$, observe e^t_m, create rl^t_m, sl^t_m and add to memory
2: **while** true **do**
3: Take action $\mathbf{a}^t_m = (1-\eta)\psi^t_m + \eta\zeta^t_m$
4: Receive P^t_{-m}, calculate u^t_m, observe $\rho^{t+1}_m, \mathbf{e}^{t+1}_m$
5: Create and add state to RL memory: rl^{t+1}_m
6: Create and add state to SL memory: $(\text{sl}^{t+1}_m, \mathbf{a}^t_m)$
7: Construct S^t_m, S^{t+1}_m, calculate $\zeta^{t+1}_m = \text{RL}(S^t_m, S^{t+1}_m, u^t_m)$
8: Calculate $\psi^{t+1}_m = \text{SL}(\text{sl}^{t+1}_m)$
9: $t \leftarrow t+1, \eta \leftarrow 1/t, \zeta^t_m \leftarrow \zeta^{t+1}_m, \psi^t_m \leftarrow \psi^{t+1}_m$
10: **end while**

Algorithm 2: RL algorithm for bidder m

1: Initialize θ, w arbitrarily. Initialize λ
2: **while** true **do**
3: Input t and S^t_m, S^{t+1}_m constructed from RL memory
4: Run critic and get $\hat{V}(S^t_m, \mathbf{w}), \hat{V}(S^{t+1}_m, \mathbf{w})$
5: Calculate $\bar{u}_m = \lambda\bar{u}_m$ and δ (immediate payoff u is reward)
6: Run actor and get $\mu(\theta), \Sigma(\theta)$
7: Sample ζ^{t+1}_m from $F(\mu, \Sigma)$, update $\mathbf{w}$ and θ
8: **end while**

4.1 Short-Term Algorithm

The short-term algorithm is based on the FSP method, it addresses the convergence challenge of a best-response algorithm. FSP balances exploration and exploitation by replaying its own past actions to learn an average behavioral strategy regardless of other bidders' strategies; then it cautiously plays the behavioral strategy mixed with best response (Heinrich et al. 2015). Table 2 summarizes the notation for this section. The method consists of two parts: a supervised learning (SL) algorithm predicts the bidder's own behavioral strategy ψ, and an RL algorithm predicts its best response ζ to other bidders. The bidder has η, $\lim_{t\to\infty}\eta = 0$ probability of choosing action $\mathbf{a} = \zeta$, otherwise it chooses $\mathbf{a} = \psi$. The action includes backoff decision α and bidding price b. If α is above a threshold, the bidder submits the bid; otherwise, the bidder backs off for a duration linear in α. We predefine the threshold to influence bidder behavior: with a higher threshold, the algorithm becomes more conservative and tends to back off more bids. Learning this threshold (e.g. through meta-learning algorithms) is left for future work.

Although FSP only converges in certain classes of games (Leslie et al. 2006) – and in our case of a multi-player, general-sum game with infinite strategies, it does not necessarily converge to an NE – it is still an important experiment as our application belongs to a very general class of games; empirical results show that by applying FSP, overall performance is greatly improved compared to using only RL. The FSP is described in Alg. 1.

Input to SL includes bidder m's current bidder information ρ^t_m (e.g. initial conditions, current reserve pool, etc.), and environment information visible to m, denoted e^t_m (e.g. number of bidders in the network, number of active bids, final price in the previous round, etc.). SL infers behavioral strategy ψ^t_m. The input $\text{sl}^t_m = (\rho^t_m, e^t_m)$ and actual action $\mathbf{a}^t_m$

Table 3: Sec.4.2 symbol definition

Sym	Description	Sym	Description
φ	featurized state	ϵ	credit assign. weight
r_i	intrinsic reward	r_e	extrinsic reward
L_f	forward mdl loss	L_i	inverse mdl loss
ξ	reward weight		

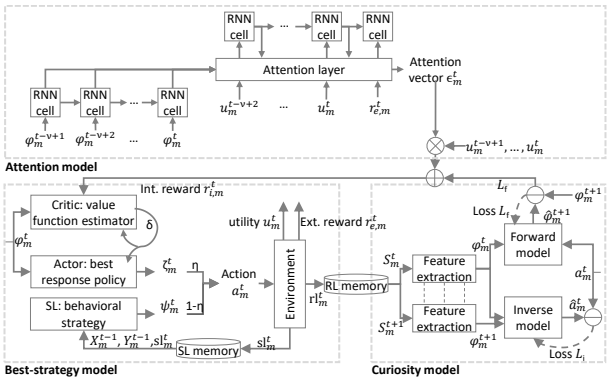


Figure 2: Long-term algorithms

are stored in SL memory to train the regression model. We use a multilayer perceptron in our implementation.

Input to RL is constructed from m's present state rl_m^t. rl_m^t is a combination of ρ_m^t; e_m^t; previous other bidders' state P_{-m}^{t-1}, represented by the final price p, or $P_{-m}^t = \mathbf{p}^t = \{p_k^t | k \in K\}$; and calculated immediate payoff u_m^{t-1}. To consider historical records, we take ν most recent states to form the complete state input to RL: $S_m^t = \{\text{rl}_m^\tau | \tau = t-\nu+1, \cdots, t\}$. RL outputs best response ζ_m (Fig. 1). We provide a detailed description of the RL algorithm below.

The RL Algorithm Our approach is similar to (Khaledi et al. 2016) in the use of a learning algorithm for the bidders to adjust their bidding price based on budget and observation of other bidders: we estimate other bidders' state P_{-m} from payment information and use the estimate as basis for a policy. Also, similar to their work, payment information is only from the broker to each bidder. However, our approach differs from (Khaledi et al. 2016) in several major points. We use a continuous space for bidder states (i.e. continuous value for payments). As also mentioned in (Khaledi et al. 2016), a finer-grained state space yields better learning results. We do not explicitly learn the transition probability of bidder states. Instead, we use historical states as input and directly determine the bidder's next action.

We use the actor-critic algorithm (Sutton and Barto 2018) for RL (Fig. 1 and Alg. 2). The **critic** learns a state-value function $V(S)$. Parameters of the function are learned through a neural network that updates with $\mathbf{w} \leftarrow \mathbf{w} + \gamma^w \delta \nabla \hat{V}(S, \mathbf{w})$, where γ is the learning rate and δ is the TD error. For a continuing task with no terminal state, no discount is directly used to calculate δ. Instead, the average reward is used (Sutton and Barto 2018): $\delta = u - \bar{u} + \hat{V}(S', \mathbf{w}) - \hat{V}(S, \mathbf{w})$. In our case, the reward is utility u. We use exponential moving average (with rate λ) of past rewards as $\bar{u}$.

Algorithm 3: Curiosity learning algorithm

1: Initialize model parameters and ϵ arbitrarily. Initialize ξ
2: **while** true **do**
3: Input a_m^t and S_m^t, S_m^{t+1} constructed from RL memory
4: Run feature extraction and get ϕ_m^t, ϕ_m^{t+1}
5: Run forward model, get $\hat{\phi}_m^{t+1}$, calculate L_f
6: Run inverse model, get $\hat{a}_m^t$, calculate L_i
7: Update forward, inverse, feature extraction model params
8: Infer from credit assignment, extract ϵ_m^t from attention layer
9: Calculate and output $r_{i,m}^t$
10: **end while**

The **actor** learns the parameters of the policy π in a multi-dimensional and continuous action space. Correlated back-off and bidding price policies are assumed to be normally distributed: $F(\mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu))$. For faster calculation, instead of covariance Σ, we estimate lower triangular matrix L ($LL^T = \Sigma$). Specifically, the actor model outputs the mean vector μ and the elements of L. Actor's final output ζ is sampled from F through $\zeta = \mu + L\mathbf{y}$, where μ is the mean and $\mathbf{y}$ is an independent random variable from standard normal distribution. Update function is $\theta \leftarrow \theta + \gamma^\theta \delta \nabla \ln \pi(\mathbf{a}|S, \theta)$. We use $\frac{\partial \ln F}{\partial \mu} = \Sigma(\mathbf{x}-\mu)$ and $\frac{\partial \ln F}{\partial \Sigma} = \frac{1}{2}(\Sigma(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T\Sigma - \Sigma)$ for back-propagation.

The objective is to find a strategy that, given input S_m^t, determines $\mathbf{a}$ to maximize $\frac{1}{T-t}\mathbb{E}[\sum_{t'=t}^T u_m^{t'}]$. To implement the actor-critic RL, we use a stacked convolutional neural network (CNN) with highway (Srivastava et al. 2015) structure similar to the discriminator in (Yu et al. 2017) for both actor and critic models. The stacked CNN has diverse filter widths to cover different lengths of history and extract features, and it is easily parallelizable, compared to other sequential networks. Since state information is temporally correlated, such a sequential network extracts features better than multilayer perceptrons. The highway structure directs information flow by learning the weights of direct input and performing non-linear transform of the input.

4.2 Long-Term Learning Algorithm

Our core contribution is the long-term reward maximization algorithm called DRACO2. It is based on the short-term RL algorithm from the previous section. We add the following features: 1) reward prediction, 2) more exploration in the early stages of learning, and 3) short- and long-term reward alignment through credit assignment. Points 1 and 2 are achieved through an adapted curiosity model. Point 3 is achieved through a hierarchical structure. The structure uses an attentional network that is responsible for learning to assign weights to short-term rewards based on their relevance to the long-term, sparse extrinsic reward, the learning process is only triggered when a new extrinsic reward becomes available (Fig. 2). Between the extrinsic reward signals, the underlying RL+curiosity model learns to better predict next states, actions, and intrinsic rewards. Table 3 summarizes the notation for this section. Next, we describe the curios-

ity learning and credit assignment models in detail.

Curiosity Model Our curiosity model is based on the vanilla model from (Pathak et al. 2017). The original model uses a feature extraction model to identify features that can be influenced by the agent's actions, thus improving the model's generalization properties in new environments. In our competitive and dynamic environment, next state depends not only on the current state, but on a number of historical states. We therefore extract features from S_m^t. The resulting featurized state vector $\phi_m^t = \text{feature}(S_m^t)$ replaces S_m^t in the previous short-term algorithm, to become the input of both the actor and the critic models, as well as the credit assignment model.

The original curiosity model uses a forward model and an inverse model to predict next state and next action, respectively. These are supervised learning models with the objective to minimize loss $L_f = \|\phi_m^t - \hat{\phi}_m^t\|_2^2$ and $L_i = \|\mathbf{a}_m^t - \hat{\mathbf{a}}_m^t\|_2^2$. One of the objectives of the forward and inverse models is to improve prediction accuracy of the consequence of the agent's actions, even without any reward signal. In our game setup, we have short-term intrinsic reward signals (only not aligned and potentially conflicting with the extrinsic rewards); therefore, we adapt the input to include the previous intrinsic reward values, and the forward model's objective is to improve prediction accuracy of both the state and the intrinsic reward.

In the original curiosity model, the intrinsic reward is the loss of the forward model: $r_{i,m}^t = \xi L_f$, and the bigger the forward loss, the higher the intrinsic reward. Through the adversarial design, the model is encouraged to explore state-actions where the agent has less experience and prediction accuracy is low. The intrinsic rewards are inserted between sparse extrinsic rewards to improve learning efficiency despite the sparseness – the authors of (Pathak et al. 2017) call this internal motivation "curiosity-driven exploration". In our approach, we apply the same method with a modified intrinsic reward definition: $r_{i,m}^t = \xi L_f^t + (1-\xi)\epsilon u_m^t$, where ξ is a pre-defined weight factor to balance between the two short-term objectives, and ϵ is a weight factor from the credit assignment model (see below). The objective is to maximize: $\mathbb{E}_\pi[\sum_t r_{i,m}^t] - L_i - L_f$. The modified long-term learning algorithm based on the short-term algorithm is in Alg. 3. Note that the FSP and Actor-Critic parts are the same as in Alg. 1 and 2, except the input to both actor and critic is the featurized state vector ϕ_m^t, instead of the original state vector S_m^t, and reward is $r_{i,m}^t$ instead of u_m^t.

Credit Assignment Model The credit assignment model uses a sequential network (recurrent neural network as encoder and decoder) with an attention layer. Typically, such a sequential network is used to identify correlation between sequenced input elements enc_i and predict a corresponding sequence of output elements $\hat{\text{dec}}_o$. The sequential network can be enhanced with an attention layer. Our credit assignment model is inspired by (Ferret et al. 2020), our model is different in that we do not decompose the extrinsic reward.

In our credit assignment model, we are not interested in predicting $\hat{\text{dec}}_o$. Instead, we want to determine the contribution of each state-action pair towards the final extrinsic reward $r_{e,m}^t$. Therefore, we trigger the training of the credit assignment model only when there is a new signal $r_{e,m}^t$ at time t: this signal becomes the last element of the target vector. We train the model on the batch of ν featurized state vectors $\text{enc}_i = \{\phi_m^{t-\nu+1}, \cdots, \phi_m^t\}$ with both short- and long-term rewards as target vector, $\text{dec}_o = \{u_m^{t-\nu+2}, \cdots, u_m^t, r_{e,m}^t\}$. In time step $\tau \in [t-\nu+1, t]$, the attention layer generates a weight vector corresponding to input vector enc_i, marking its relevance to the current output prediction $\hat{\text{dec}}_o^\tau$, until in the last time step t, the attention layer outputs a weight vector $\epsilon_m^t = \{\epsilon_1, \cdots, \epsilon_\nu | \sum_{i=1}^n \epsilon_i = 1\}$ corresponding to enc_i that marks their relevance to the last output $r_{e,m}^t$. Model parameters are updated with the mean square error between the generated output $\hat{\text{dec}}_o$ and target vector dec_o.

The weight vector ϵ_m^t is then multiplied with the original auction payoffs u_m^t. Through ϵ_m^t, short- and long-term rewards are aligned, even if they are conflicting in nature. Between sparse extrinsic rewards, only the forward network of credit assignment model is run to infer a weight vector.

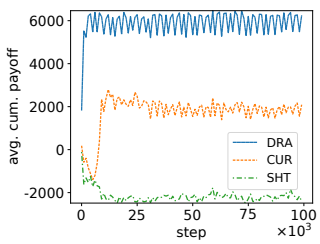
Algorithm 4: Credit assignment algorithm

```
Initialize model parameters arbitrarily, initialize batch size ν
Input r_{e,m}^t and S_m^t, ..., S_m^{t-ν+1}, u_m^t, ..., u_m^{t-ν+2} from RL memory
Run feature extraction and get φ_m^t, ..., φ_m^{t-ν+1}
for τ ← t − ν + 1 to t − 1 do
    Input φ_m^τ to encoder, get encoder output enc_o
    Input enc_o, u_m^{τ+1} to decoder, get output dec_o^τ
end for
Input φ_m^t to encoder, get enc_o
Input enc_o, r_{e,m}^t to decoder, get dec_o^t
Update model parameters, output ε_m^t from attention layer
```

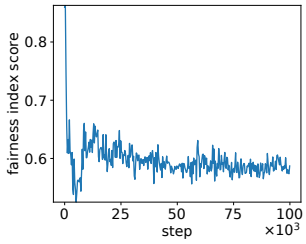
5 Evaluation

We train DRACO2 in two repeated auction games with a Python discrete event simulator. Both games have six bidder agents and one broker agent. The broker is a passive agent without learning capabilities. In every time step, the broker offers one commodity (e.g. object or service) for bidding, all bidders can join the auction simultaneously. The broker grants the commodity to the bidder with the highest score; ties are broken randomly. An immediate payoff is given to the winner, the value of the payoff is specific to the type of game. Except the winner, all other participating bidders pay a fixed cost for joining the auction.

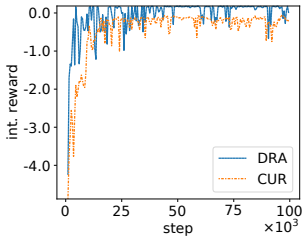
All bidders start with a reserve pool of wealth; it is updated every time step with payoffs and costs. Regardless of the bidder's behavior, there is a constant cost each time step (carrying cost). If the pool is depleted, the game is over for the bidder, it receives a penalty, and rejoins the game with the same initial reserve. Otherwise, the game continues for a certain number of time steps (in our simulation we take $T = 150$ time steps). When the game ends, all bidders restart the game with the same initial reserve. In the case of long-term learning algorithms, bidder m receives a long-term extrinsic reward signal at the end of each game. It can



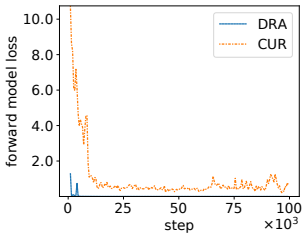
(a) Payoff-performance per agent, by algorithm type.



(b) Fairness-performance.

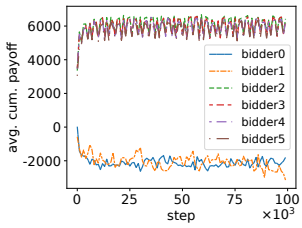


(c) Intrinsic reward, comparison of DRA and CUR agents.

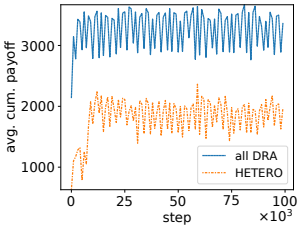


(d) Forward model loss, comparison of DRA and CUR agents.

Figure 3: FP with HETERO agents and payoff-signal



(a) Payoff-performance: 4/6 agents (bidders2-5) receive max. reward.



(b) Payoff-performance comparison: all DRA vs. HETERO.

Figure 4: FP with only DRA agents and payoff-signal

be m's own cumulated payoff in the reserve pool of wealth: $r^t_{e,m} = U^t_m = \sum_{t-T}^{t} u^\tau_m$, or overall fairness, defined as the J-index (Jain et al. 1984) of payments from the broker to the bidder agents over time: $r^t_{e,m} = \frac{(\sum_m \sum_{t-T}^{t} p^t_m)^2}{|M| \sum_m (\sum_{t-T} t p^t_m)^2}, \forall m \in M$, which is commonly used to measure fairness in networking. J-index is the reciprocal of the original normalized Herfindahl–Hirschman Index (Rhoades 1993) used to measure market concentration. To preserve privacy, extrinsic reward signals do not contain private agent information. There are free and occupied bidders: if a bidder wins a bid, its resources are occupied for a period of time, i.e. service duration, during which the occupied bidder cannot submit new bids. Each free bidder decides on 1) whether to join the auction for the commodity in the current time step, 2) if so, a bidding price b that is lower than or equal to the amount in the reserve pool of wealth, and 3) other decision factors required by the specific auction setup. In the first-price reverse auction, service duration d is correlated with the bidding price b. The broker gives bidders a balanced score based on the multiplication of price and service duration. Winner of the auction gets a immediate payoff of $b \cdot d$. In the second-price forward auction, bidders decide on bidding price b, and d is a constant value (to simplify winning criterion and the calculation of second-price). If it wins, the bidder pays the broker the second-highest bidding price p among all bidders. The winner gets an immediate payoff of $(b - p) \cdot d$. In both games, during the service duration d, the winner cannot join any new auctions.

The bidders may use one of three learning algorithms: the short-term algorithm (SHT), the long-term algorithm based on curiosity learning (CUR), and DRACO2, the long-term algorithm with an attention layer for credit assignment (DRA). In the setup with homogeneous agents, all six agents have an algorithm of the same type. In the setup with heterogeneous agents, each algorithm is given to two bidders, and all algorithms compete in the same auction game.

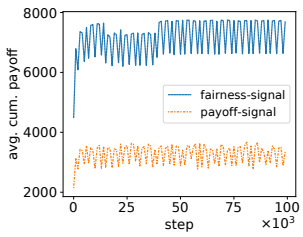
To summarize, we simulate first-price reverse (FP) or second-price forward (SP) auction, with homogeneous (DRA, CUR or SHT) or heterogeneous agents (HETERO), and use either average cumulated payoff per agent (payoff-signal), or fairness index score (fairness-signal) as extrinsic reward signals. As performance measure we measure the average cumulated payoff per agent (payoff performance), and the overall fairness index score (fairness performance). All results come from continuous training.

5.1 First-Price Reverse Auction (FP)

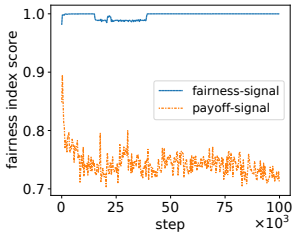
First-price reverse auction (lowest-bid-wins) is common e.g. in long-term energy contracts (Lucas et al. 2013) or network resource allocation (Xu et al. 2012b) where multiple resource owners bid to sell to one buyer that prefers low price for a long duration.

Each curve in Fig. 3a represents the average performance of two agents with the same type of algorithm, in a heterogeneous setting. Both DRA and CUR agents outperform SHT: through the reserve pool of wealth, current behavior influences bidding decisions in the future and has direct impact on the delayed extrinsic reward. However, the short-term algorithm values the immediate intrinsic reward much higher than the extrinsic reward in the distant future, therefore failing to compete in the game. On the other hand, the DRA agents clearly performs the best, but at the cost of other agents with less aggressive algorithms, as is shown by the low fairness index in Fig. 3b. Figures 3c and 3d compare training performance of DRA and CUR agents in the game. The DRA agent not only converges faster, it also converges to a lower loss and higher intrinsic reward.

If we pitch the aggressive DRA agents against each other, i.e. all six agents are DRA agents, we have similar a result (Fig. 4a): only four DRA agents can maximize their cumu-

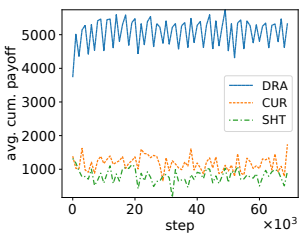


(a) Payoff-performance comparison: payoff-signal vs. fairness-signal.

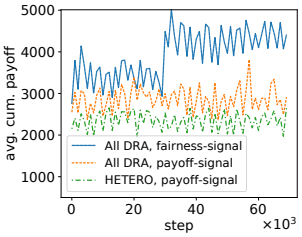


(b) Fairness-performance comparison: payoff-signal vs. fairness-signal.

Figure 5: FP with only DRA agents and fairness-signal



(a) Payoff-performance: HETERO w/ payoff-signal, by algorithm type.



(b) Payoff-performance comparison.

Figure 6: SP, all DRA vs. HETERO, payoff-signal vs. fairness-signal

lated payoff over time, although the game has a higher social welfare, compared to the HETERO case (Fig. 4b). The difference in individual performance is caused by DRA agents' aggressive, selfish (i.e. with private indiviual goals), rational (i.e. act to maximize reward) behavior. They profit from an unregulated system at the cost of social welfare. In fact, it is possible for all six agents to maximize their reward: to motivate cooperation, we replace the cumulated payoff with fairness index score as long-term extrinsic reward signal. The negative impact on social welfare can thus be prevented.

Fig. 5 compares two independent simulation results. The dotted orange curve is the average cumulative payoff of six DRA agents when the extrinsic reward is also the cumulative payoff. The solid blue curve is when the extrinsic reward is fairness index score. With fairness as incentive, all agents receive better cumulated payoffs, while achieving a much higher fairness index score. Hence, with our solution, it is possible to increase both individual gain and social welfare.

To wrap up, the first simulation setup (Fig. 3) demonstrates how the DRACO2 algorithm learns quickly and aggressively in a multi-agent, dynamic environment with partial information, a big state-action space, and sparse / delayed extrinsic reward. The second setup (Fig. 5) demonstrates how DRACO2 can be easily optimized to integrate a system goal while preserving privacy and individual goals.

5.2 Second-Price Forward Auction (SP)

In a second-price forward auction (second-highest-bid-wins), the broker is a seller that grants the commodity to the bidder with highest bidding price, but the payment for the commodity is the second-highest price of all bidding prices. This type of auction is common for selling public goods, maximizing welfare rather than seller profit, e.g. in networking resource allocation (Xu et al. 2012a) and e-commerce (Huang and Kauffman 2011), where multiple end users bid for resources from one service provider.

Fig. 6 shows similar results in SP as in FP. When three types of agents co-exist in a profit-oriented setup, the two DRA agents win at the cost of social welfare (Fig. 6a). Social welfare increases when all agents are DRA agents. Finally, if the DRA agents are instead given a fairness index as incentive, social welfare reaches is much higher. This can be seen from Fig. 6b: with fairness index score as extrinsic reward signal, social welfare increases.

6 Conclusion

We demonstrate the performance of DRACO2 in two repeated auction games. The results show that, with the help of an attention layer for long-term credit assignment, the DRA agents behave more aggressively in the competition against other agents, when the long-term goal is to maximize cumulated private payoff. However, the selfish behavior has a negative impact on the overall social welfare. To encourage cooperation, we use fairness as the long-term goal. Simulation results show the improvement in individual payoff and in overall fairness index score.

We ran the simulations with only six agents, and the simulated auction mechanisms are relatively simple. In the next steps, we would focus on increasing the number of agents in the simulation, give them different individual goals, and test the algorithm in more complex setups, especially with more realistic extrinsic reward signals.

References

A, T.; et al. 2018. Learning from delayed outcomes with intermediate observations. *arXiv preprint arXiv:1807.09387*.

Almasri, M.; et al. 2020. Dynamic decision-making process in the opportunistic spectrum access. *Advances in Science, Technology and Engineering Systems Journal*.

Bowling, M.; et al. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence*.

Burda, Y.; et al. 2019. Large-Scale Study of Curiosity-Driven Learning. In *ICLR*.

Busoniu, L.; et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*.

Chang, Y.-H. 2007. No regrets about no-regret. *Artificial Intelligence*.

Dulac-Arnold, G.; et al. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 1–50.

Einav, L.; et al. 2018. Auctions versus posted prices in online markets. *Journal of Political Economy*.

Feigenbaum, J.; et al. 2007. Distributed algorithmic mechanism design. In *Algorithmic Game Theory*. Cambridge University Press.

Ferret, J.; et al. 2020. Self-Attentional Credit Assignment for Transfer in Reinforcement Learning. In *IJCAI*.

Heinrich, J.; et al. 2015. Fictitious self-play in extensive-form games. In *ICML*.

Hester, T.; and Stone, P. 2013. Texplore: real-time sample-efficient reinforcement learning for robots. *Machine learning*.

Huang, H.; and Kauffman, R. J. 2011. On the design of sponsored keyword advertising slot auctions: An analysis of a generalized second-price auction approach. *Electronic Commerce Research and Applications*.

Hung, C.-C.; et al. 2019. Optimizing agent behavior over long time scales by transporting value. *Nature communications*.

Jain, R. K.; et al. 1984. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory*.

Khaledi, M.; et al. 2016. Optimal bidding in repeated wireless spectrum auctions with budget constraints. In *IEEE GLOBECOM*.

Leslie, D. S.; et al. 2006. Generalised weakened fictitious play. *Games and Economic Behavior*.

Lucas, H.; et al. 2013. Renewable energy auctions in developing countries. *International Renewable Energy Agency*.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *Machine learning proceedings*.

Pathak, D.; et al. 2017. Curiosity-driven exploration by self-supervised prediction. In *ICML*.

Rhoades, S. A. 1993. The herfindahl-hirschman index. *Fed. Res. Bull.*

Schindler, R. M.; et al. 2011. *Pricing strategies: a marketing approach*. sage.

Shahriari, B. 2017. *Generic Online Learning for Partial Visible & Dynamic Environment with Delayed Feedback*. Ph.D. thesis, San Jose State University.

source. 2021. https://github.com/DRACOsource/biddinggame.

Srivastava, R. K.; et al. 2015. Training very deep networks. In *NeurIPS*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Teng, Y.; et al. 2013. Reinforcement-learning-based double auction design for dynamic spectrum access in cognitive radio networks. *Wireless Personal Communications*.

Weinberg, M.; et al. 2004. Best-response multiagent learning in non-stationary environments. In *AAMAS*.

Xu, C.; et al. 2012a. Interference-aware resource allocation for device-to-device communications as an underlay using sequential second price auction. In *IEEE ICC*.

Xu, C.; et al. 2012b. Resource allocation using a reverse iterative combinatorial auction for device-to-device underlay cellular networks. In *IEEE GLOBECOM*.

Yu, L.; et al. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.