Correlated Learning for Aggregation Systems

Tanvi Verma and **Pradeep Varakantham**
School of Information Systems, Singapore Management University
tanviverma.2015@phdis.smu.edu.sg, pradeepv@smu.edu.sg

Abstract

Aggregation systems (e.g., Uber, Lyft, FoodPanda, Deliveroo) have been increasingly used to improve efficiency in numerous environments, including in transportation, logistics, food and grocery delivery. In these systems, a centralized entity (e.g., Uber) aggregates supply and assigns them to demand so as to optimize a central metric such as profit, number of requests, delay etc. Due to optimizing a metric of importance to the centralized entity, the interests of individuals (e.g., drivers, delivery boys) can be sacrificed. Therefore, in this paper, we focus on the problem of serving individual interests, i.e., learning revenue maximizing policies for individuals in the presence of a self interested central entity. Since there are large number of learning agents that are homogenous, we represent the problem as an Anonymous Multi-Agent Reinforcement Learning (AyMARL) problem. By using the self interested centralized entity as a correlation entity, we provide a novel learning mechanism that helps individual agents to maximize their individual revenue. Our Correlated Learning (CL) algorithm is able to outperform existing mechanisms on a generic simulator for aggregation systems and multiple other benchmark Multi-Agent Reinforcement Learning (MARL) problems.

1 INTRODUCTION

In many real-world domains, there is a need to match supply with customer demand continuously. For example, taxi aggregation companies match taxis to customers, food delivery aggregation companies match restaurants and delivery boys to customers. We refer to these problems as Multi-agent Sequential Matching Problems (MSMPs). In these MSMPs, individuals (taxi drivers, delivery boys etc.), who provide supply, earn more by being at advantageous locations. In this work we develop an approach for individual agents to learn these advantageous locations in the presence of other learning agents.

Existing research has represented such learning problems, where learning agents have to learn in the presence of other learning agents using the Multi-Agent Reinforcement Learning (MARL) model. MARL problems are challenging as interaction between agents makes the environment non-stationary and hence Q-learning based approaches (which rely on domain stationarity) typically do not converge.

Existing work [Littman, 1994; Hu et al., 1998; Hu and Wellman, 2003] has focussed on computing equilibrium policies by representing MARL problems as learning in Stochastic Games (SG). Due to the existence of multiple equilibria and the challenge of coordinating agents to focus on the same equilibria, other alternatives have been considered [Shoham et al., 2003; Weinberg and Rosenschein, 2004]. Bowling and Veloso [2001] proposed the following criteria for multi-agent learning: (1) *rationality*: learning should terminate with a best response to the play of other agents and (2) *convergence*: learning should converge to a stationary policy. In this paper, we focus on these two criterion for MARL problems with a large number of homogenous agents.

The key contribution of this paper is in developing a generic learning approach that exploits the presence of an aggregation system (e.g., Uber, Lyft, FoodPanda). We propose Correlated Learning (CL), where individual agents learn to play best response against a central agent, which learns joint actions that maximize social welfare. Similar to the work by Bowling and Veloso [2001], we demonstrate that CL satisfies the *rationality* and *convergence* criteria when the agent population is large.

We empirically show on multiple MARL problems that CL results into a "win-win situation" where both central agent and individual agents receive better payoff than the other MARL algorithms suitable for individual learning in the presence of a large number of agents. For aggregation systems, where there are many similar agents, we consider Anonymous MARL model that can capture homogeneity in agent models and anonymity in agent interactions to ensure scalable and efficient learning.

2 MOTIVATING PROBLEMS: MSMPs

This paper is motivated by Multi-agent Sequential Matching Problems (MSMPs) where there are multiple agents and there is a need for these agents to be matched to customer demand. Aggregation systems (Uber, Lyft, Deliveroo etc.) maximize the overall system wide revenue in MSMPs. A key characteristic of these domains is that interactions between individual agents are anonymous. Here we describe three popular MSMPs:

Taxi Aggregation: Companies like Uber, Lyft, Didi, Grab etc. provide taxi supply aggregation systems. The goal is to ensure wait times for customers is minimal or amount of revenue earned is maximized by matching taxi drivers to customers. However, these goals of the aggregation companies may not be aligned to the individual driver objective of maximizing their own revenue. The method provided in this paper will be used to guide individual drivers to "right" locations at "right" times based on their past experiences of customer demand, taxi supply and guidance provided by the aggregation companies. Interactions between taxi drivers are anonymous, because the probability of a taxi driver being assigned to a customer is dependent on the number of other taxi drivers being in the same zone (and not on specific taxis being in the zone) and customer demand.

Food or Grocery Delivery: Aggregation systems have also become very popular for food delivery (Deliveroo, Ubereats, Foodpanda, DoorDarsh etc.) and grocery delivery (AmazonFresh, Deliv, RedMart etc.) services. They offer access to multiple restaurants/grocery stores to the customers and use services of delivery boys/delivery vans to deliver the food/grocery. Similar to taxi case, there is anonymity in interactions as the probability of a delivery boy/van being assigned a job is dependent on number of other delivery boys/vans being in the same zone and customer demand.

Supply Aggregation in Logistics: More and more online buyers now prefer same day delivery services and tradition logistic companies which maximize usage of trucks, drivers and other resources are not suitable for it. Companies like Amazon Flex, Postmates, Hitch etc. connect shippers with travelers/courier personnel to serve same day/on-demand delivery requests. The courier personnel in this system can employ the proposed method to learn to be at "right" place at "right" time by learning from the past experiences. Interactions between couriers are anonymous due to dependence on number of other couriers (and not on specific couriers).

3 RELATED WORK

MARL algorithms that represent the model as stochastic games can be broadly divided into two categories: equilibrium based learning and best response based learning.

In the equilibrium based learning, algorithms try to learn policies which results into Nash equilibrium or $\epsilon-$Nash equilibrium. Minimax-Q [Littman, 1994] is considered to be the first equilibrium-based MARL algorithm which uses minimax rule to learn equilibrium policy in two-player zero-sum Markov games. Hu and Wellman [2003] proposed Nash-Q learning which extends the classic single agent Q-learning [Watkins and Dayan, 1992] to general sum stochastic games. Nash-Q learning uses value of Nash equilibria of each state to update the Q-values and is shown to converge to equilibrium under certain assumptions. Friend-or-Foe Q-learning (FFQ) [Littman, 2001] proposed learning adversarial and coordination equilibrium and it has less strict convergence condition compared to Nash-Q. Correlated-Q learning [Greenwald et al., 2003] is similar to Nash-Q but it uses value of correlated equilibria to update the Q-values instead of Nash equilibria. These algorithms are not practical in environments with large number of agents and with large state and action space. One recent example in this category is Mean field Q-learning (MFQ) [Yang et al., 2018] which where individual agents learn Q-values of its interaction with average action of its neighbour agents. These algorithms focusing on equilibrium learning does not provide any guarantee that the policy will not converge to a sub-optimal equilibrium in case multiple Nash equilibria exists. *Equilibrium selection* [Harsanyi et al., 1988] is a sub-field of game theory which focuses on selecting certain equilibrium over another. The discussion on equilibrium selection is outside the scope of this paper.

In the best response based MARL, individual agents try to learn policies which are best response to the joint policy of the other agents. NSCP (non-stationary converging policies) learning was proposed by Weinberg and Rosenschein [2004] which models other agents in the environment and learns a best response policy. Lanctot et al. [2017] uses deep reinforcement learning to compute the best response to a distribution over policies, but it assumes prior knowledge of a set of opponent policies. Learning with Opponent Learning Awareness (LOLA) was introduced by Foerster et al. [2018] which minutely modifies the objective of the player to take into account their opponents' goals. Though these algorithms which model opponents are relevant to our work, they do not scale to a large number of agents present in aggregation systems. Fictitious self play (FSP) [Heinrich et al., 2015] is an excellent example of learning of best response through self play. FSP is a machine learning framework that implements fictitious play [Brown, 1951] in a sample-based fashion. Heinrich and Silver [2016] proposed neural fictitious self play (NFSP) which combines FSP with neural network function approximator. Learning best response through self play in MARL are known to be scalable. However, they are often sub-optimal because environment becomes non-stationary from a single agent's perspective. While learning best responses, some recent work [Lowe et al., 2017; Nguyen et al., 2017; Yang et al., 2018] consider presence of a central agent which provides extra information to the individual agents. Our work uses the similar framework but we also consider the fact that the objective of the central agent might not be aligned with the objective of individual agents. More specifically, in CL individual agents learn to play best response policy to the central agent's social welfare policy.

4 BACKGROUND

In this section we provide a brief overview of reinforcement learning and other relevant concepts.

4.1 REINFORCEMENT LEARNING (RL)

Reinforcement Learning [Sutton and Barto, 1998] is a popular method for solving Markov Decision Process (MDP) when the model of MDP is not known. An MDP is formally defined as the tuple $\langle S, A, T, R\rangle$, where S is the set of states, A is the set of actions, $T(s, a, s')$ represents the probability of transitioning from state s to state s' on taking action a and $R(s, a)$ represents the reward obtained on taking action a in state s. RL agents learn a policy that maximizes their expected future reward while interacting with the environment. Q-learning [Watkins and Dayan, 1992] is one of the most popular RL approach, where the value function $Q(s, a)$ are updated based on experiences given by (s, a, s', r):

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$

Where α is the learning rate and γ is the discount factor.

DQN Mnih et al. [2015] approximates the Q-values with a deep neural network. This deep network for Q-values is parameterized by a set of parameters, θ and the parameters are learned using an iterative approach that employs gradient descent on the loss function. Specifically, the loss function at each iteration is defined as follows:

$$\mathcal{L}_\theta = \mathbb{E}[(y^{DQN} - Q(s, a; \theta))^2] \tag{1}$$

where $y^{DQN} = r + \gamma \text{max}_{a'} Q(s', a'; \theta^-)$ is the target value computed by a target network parameterized by previous set of parameters, θ^-. Parameters θ^- are frozen for some time while updating the current parameters θ. To ensure independence across experiences this approach maintains a replay memory $\mathcal{J}$ and then samples experiences from that replay memory.

4.2 ANONYMOUS MULTI-AGENT REINFORCEMENT LEARNING (AyMARL)

Multi-Agent RL (MARL) involves multiple agents in a shared environment which must learn to maximize either joint payoff (cooperative MARL) or their individual payoffs (competitive/non-cooperative MARL). For MSMPs, the environment is generally considered to be divided into $\mathcal{Z}$ zones and action of individuals is to select a zone to move. Verma et al. [2019] modeled anonymous version of MARL (AyMARL) for MSMPs. AyMARL is a specialization of the SG model that considers interaction anonymity. Formally, it is represented using the tuple:

$$\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i \in \mathcal{N}}, \mathcal{T}, \{\mathcal{R}_i\}_{i \in \mathcal{N}} \rangle$$

$\mathcal{N}$ is the number of agents. $\mathcal{S}$ is the set of states, which is factored over individual agent states in MSMPs. For instance in the taxi problem of interest, an environment state **s** is the vector of locations (zone) of all agents, i.e., $\mathbf{s}(\in \mathcal{S}) = (z_1, \ldots, z_i, \ldots, z_{\mathcal{N}})$. $\mathcal{A}_i$ is the action set of each individual agent. In case of the taxi problem, an action $a_i(\in \mathcal{A}_i)$ represents the zone to move to if there is no customer on board. $\mathcal{T}$ is the transitional probability of environment states given joint actions.

Given interaction anonymity of the agents, the state and action space are further compacted as agent population distribution over zones and aggregated action distribution of agents. More specifically, state of the environment is represented as $\boldsymbol{s} = (d_1, \ldots, d_{\mathcal{Z}})$, where d_z is the fraction of agent population present in zone z, i.e., $\sum_{z \in \mathcal{Z}} d_z = 1$. Furthermore, the action available to agents present in zone z is $\mathcal{A}_z$, the set of zones which can be reached from z in a single time step. The joint action is then defined as $\boldsymbol{a} = (\boldsymbol{a}^1, \ldots, \boldsymbol{a}^{\mathcal{Z}})$, where $\boldsymbol{a}^z = (a_z^k)_{k \in \mathcal{A}_z}$ and a_z^k is the fraction of agents present in zone z selecting to move to zone k ($\sum_{k \in \mathcal{A}_z} a_z^k = 1$).

5 CORRELATED LEARNING FOR HOMOGENEOUS AGENTS

To understand the core facets of the approach, we first develop correlated learning method for homogenous agents in this section. We extend it to AyMARL problems for MSMPs in the next section. As discussed in the Section 1, the objective of the central agent is to maximize the social welfare whereas individual agents try to maximize their own payoff.

There are $\mathcal{N}$ individual agents and one central agent present in the environment. Central agent does not interact directly with the environment and learns only from the experiences of the individual agents. Just to reiterate, $a_i \in \mathcal{A}$ denote action of agent i, where $\mathcal{A}$ is the action space of the individual agents. As agents are homogeneous, instead of modeling their joint action as a vector of actions of individual agents, we model it as a vector of number of agents selecting each available action. We use $\boldsymbol{a} = (n_j)_{j\in\mathcal{A}}, \boldsymbol{a} \in \boldsymbol{\mathcal{A}}$ to denote the joint action where n_j is the number of agents selecting action $j \in \mathcal{A}, \sum_{j\in\mathcal{A}} n_j = \mathcal{N}$ and $\boldsymbol{\mathcal{A}}$ is the joint action space. Superscript c is used to denote that action $\boldsymbol{a}^c$ has been derived from the central agent's policy. $\boldsymbol{a}_{-i}$ is the joint action of all the agents except agent i, i.e., $\boldsymbol{a} = (\boldsymbol{a}_{-i}, a_i)$. Note that as $\boldsymbol{a}$ is count based and does not consider agents' identities, there are $|\mathcal{A}|$ possible combinations of $\boldsymbol{a}_{-i}$ and a_i which will produce a single joint action $\boldsymbol{a}$. State space is denoted by $\mathcal{S}$.

Algorithm 1 provides the key four steps involved in CL.

- Central agent suggests current social welfare policy to all the individual agents. Line 4 of the algorithm shows this step.
- Individual agents either follow the suggested action with ϵ_2 probability or play their best response policy with $1-\epsilon_2$ probability. While playing the best response policy, the individual agents explore with ϵ_3 probability (i.e. ϵ_3 fraction of $(1-\epsilon_2)$ probability) and with the remaining probability ($(1-\epsilon_3)$ fraction of $(1-\epsilon_2)$) they play their best response action. Line 6 shows this step.
- Environment moves to the next state. All the individual agents observe their individual reward and update their best response values assuming that the other agent followed the suggested action. Line 7 shows this step.
- Central agent observes the true-joint action performed by the individual agents. Based on the cumulative reward of all the individual agents and the true joint-action, the central agent updates its own learning. Line 8 shows this step.

Algorithm 1 Correlated Learning

```
1: Initialize central agent's Q-values arbitrarily, $Q_c(s, \boldsymbol{a}) \forall s \in \mathcal{S}, \forall \boldsymbol{a} \in \boldsymbol{\mathcal{A}}$
2: For all the individual agent $i$, initialize best response Q-values arbitrarily $Q_i(s, \boldsymbol{a}_{-i}, a_i) \forall s \in \mathcal{S}, \forall(\boldsymbol{a}_{-i}, a_i) \in \boldsymbol{\mathcal{A}}$
3: while not converged do
4:     compute joint action for the central entity
           $\boldsymbol{a}^c \leftarrow \epsilon_1$-greedy$\big(Q_c(s, \boldsymbol{a})\big)$.
5:     for All agent $i$ do
6:         with probability $\epsilon_2$,
               $a_i \leftarrow$ follow $\boldsymbol{a}^c$
           with remaining probability $1 - \epsilon_2$
               $a_i \leftarrow \epsilon_3$-greedy$\big(Q_i(s, \boldsymbol{a}^c_{-i}, a)\big)$
7:         Perform action $a_i$ and observe next sate $s'$ and reward $r_i$. Update agent's learning.
           $Q_i(s, \boldsymbol{a}^c_{-i}, a_i) \leftarrow (1-\alpha) Q_i(s, \boldsymbol{a}^c_{-i}, a_i) + \alpha\Big[r_i + \gamma \max_{a', \boldsymbol{a}'^c_{-i}} Q_i(s', \boldsymbol{a}'^c_{-i}, a')\Big]$
8:     Compute true joint action $\boldsymbol{a}$, central agent's reward $r_c = \sum_i r_i$. Update central agent's learning.
       $Q_c(s, \boldsymbol{a}) \leftarrow (1-\alpha) Q_c(s, \boldsymbol{a}) + \alpha\Big[r_c + \gamma \max_{\boldsymbol{a}'} Q_c(s', \boldsymbol{a}')\Big]$
```

To follow $\boldsymbol{a}^c$ is step 6, individual agents take action j with probability $n_j/\mathcal{N}$. $\boldsymbol{a}'^c$ in step 7 is the social welfare policy of central agent for state s'. The individual agents play their best response based on their belief that others are following the suggested policy whereas the central agent acts as a correlation entity, hence we call it correlated learning. There is a difference between the way individual agents and central agent update their Q-values in steps 7 and 8. While central agent updates the value based on true joint action performed, the individual agents update their values based on the recommendation of joint action by the central entity.

Central agent's learning is dependent on the experiences of the individual agents. Hence, if individual agents explore sufficiently (infinite exploration is a criteria for the learning to converge), the central agent's joint-action exploration would also be sufficient. We now argue that social welfare policy and individual agents' best response policies converge to a stationary policy.

Lemma 1. *Central agent's learning converges to a stationary policy.*

Proof. Given the ϵ-greedy approach for each of the individual agents, all the joint state and joint action combinations will be explored in the limit with infinite exploration (GLIE) [Singh et al., 2000]. Since learning of central agent is only dependent on the combined experiences of the individual agents and since individual agent exploration is exhaustive, central agent's learning is equivalent to reinforcement learning. Therefore, it will converge to a stationary policy. □

5.1 DISCUSSION

As indicated earlier, our focus is on finding rational policies that are stationary. While we cannot yet guarantee, we are able to intuitively argue for these properties.

First, as individual agents update their Q-values over individual action and joint recommended action for all other agents, it will be ideal if our approaches learn the true best response values for ensuring rational behavior. This is feasible if we generate sufficient experiences of the suggested joint action. In Algorithm 1, each individual agent either follows the suggested action, or selects a random action. Selecting random action while exploration can be considered as idiosyncratic i.i.d. (independent and identically distributed) noise. We argue that during high exploration phase they do generate experiences of the suggested joint action. The intuition comes from the common assumption in the game theory that the idiosyncratic noise is averaged away if the agent population is large [Sandholm, 2010; Carmona and Delarue, 2014; Nutz, 2018]. Hence, during exploration they generate experiences of suggested joint action and thus learn the values for true joint action.

Once the individual agents have learned values of true joint actions, the action suggested by the central agent works as a synchronization mechanism for them. Suppose for suggested joint action $\boldsymbol{a}^c$ for state s, agents have figured out their respective best response action and the resulting aggregated joint action is $\mathfrak{a}$. Even if in reality $\boldsymbol{a}^c$ and $\mathfrak{a}$ are not same, whenever joint action $\boldsymbol{a}^c$ is suggested, agents will play their respective best response and the true joint action will always be $\mathfrak{a}$. As shown in Lemma 1, the social welfare policy converges to a stationary policy, hence the best response policies will also converge to a stationary policy.

Hence under normal assumptions of Q-learning, CL intuitively satisfies the two criteria of the *rationality* and *convergence* if the agent population is large.

6 CL FOR AyMARL

While the basic version of CL described in the previous section is easy to understand, it does not scale very well due to the combinatorial state and action spaces when considering large numbers of agents. We now extend CL for large scale AyMARL problems by converting discrete combinatorial state and action spaces into continuous values. We use the zone based model of MSMPs provided in Section 4.2. As discussed in that section, both joint state and joint action are vector of continuous values (fraction of agent population), hence, a tabular learning of Q-values is difficult. Deep neural network, which is popular for function approximation, can be used to estimate $Q(\boldsymbol{s}, \boldsymbol{a})$ values for the central agent [1]. Deep deterministic policy gradient [Lillicrap et al., 2015] is an excellent algorithm to learn deterministic policies for domains with continuous actions. The assumption is that the policy is deterministic, which is a reasonable assumption for MSMPs of interest. Central agent maintains an actor network $\mu_c(\boldsymbol{s}; \theta_c^\mu)$ and a critic network $Q_c(\boldsymbol{s}, \boldsymbol{a}; \theta_c^Q)$. Central agent learns the social welfare policy from the experiences of the individual agents. The target value mentioned in Equation 1 for the central agent is computed as follows.

$$y = \sum_{i \in \mathcal{N}} r_i + \gamma Q_c(\boldsymbol{s'}, \mu_c(\boldsymbol{s'}; \theta_c^\mu); \theta_c^Q)$$

The joint state in AyMARL is aggregated distribution of agent population and from an individual agent's perspective, it needs to keep track of its exact location for efficient learning of best responses. Hence, the state view of an individual agent is given by $(\boldsymbol{s}, z_i)$ which is the current location of the agent for a given aggregated joint state $\boldsymbol{s}$. The experience of an individual agent is given by $< \boldsymbol{s}, z_i, \boldsymbol{a}_c^z, a_i, r_i, \boldsymbol{s'}, z_i' >$ which can be interpreted as after taking action a_i in state $(\boldsymbol{s}, z_i)$ agent i received r_i as immediate payoff and moved to state $(\boldsymbol{s'}, z_i')$ given the joint action suggested by the central agent was $\boldsymbol{a}_c^z$. The agents maintain their best response value network $Q_i(\boldsymbol{s}, \boldsymbol{a}_c^{z_i}, z_i, a_i; \theta_i)$ where $a_i \in \mathcal{A}_{z_i}$ is the available actions in zone z_i. Furthermore, the individual agents compute their target value as follow.

$$y_i = r_i + \gamma \max_{a_i'} Q_i(\boldsymbol{s'}, \boldsymbol{a}_c^{z_i'}, z_i', a_i'; \theta_i)$$

Here $\boldsymbol{a}_c^{z_i'}$ is the social welfare policy of the central agent for zone z_i' for the next joint state $\boldsymbol{s'}$.

7 EXPERIMENTS

We analyze and evaluate CL on four different example domains. We first experiment on a taxi simulator which is validated on a real-world data. Then we evaluate it on a synthetic online to offline service aggregation simulator. To show the effectiveness of CL, we also perform experiments on two stateless games.

[1]Use of non-linear function approximator such as neural network does not preserve any mathematical convergence guarantees. However in practice it has been seen to converge [Mnih et al., 2013, 2015].

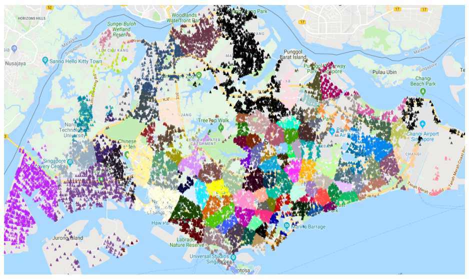


Figure 1: Road network of Singapore divided into zones

Taxi Simulator

Figure 1 shows the map of Singapore where road network is divided into zones (figure from Verma et al. [2019]). First, based on the GPS data of a large taxi-fleet, we divided the map of Singapore into 111 zones. Then we used the real world data to compute the demand between two zones and the time taken to travel between the zones. CL is scalable as it focuses on individual learning. However, simulating a very large number of agents (each of them using deep neural network) requires extensive computer resources and with the academic resources available, we could not perform a simulation with a very large number of agents. Hence, we computed proportional demand for 100 agents and simulated for 100 agents.

Synthetic Online to Offline Service Simulator

We build the synthetic simulator used in Verma et al. [2019].We generated synthetic data to simulate various combinations of demand and supply scenarios in online to offline service settings described in Section 2. We used grid world and each grid is treated as a zone. Demands are generated with a *time-to-live* value and the demand expires if it is not served within *time-to-live* time periods. Furthermore, to emulate the real-world jobs, the revenues are generated based on distance of the trip (distance between the origin and destination grids). There are multiple agents in the domain and they learn to select next zones to move to such that their long term payoff is maximized. At every time step, the simulator assigns a trip to the agents based on the number of agents present at the zone and the customer demand. In our experiments, we make a realistic assumption that the trip might not end in a single time step and time taken to complete the trip is proportional to the distance between the origin and destination grids. The revenue of an agent can be affected by features of the domain such as

- Demand-to-Agent Ratio (DAR): The average number of customers per time step per agent.
- Trip pattern: The average length of trips can be uniform for all the zones or there can be few zones which get longer trips (for ex. airports which are usually outside the city) whereas few zones get relatively shorter trips (city center).
- Demand arrival rate: The arrival rate of demand can be either static w.r.t. the time or it can vary with time (dynamic arrival rate).

We performed experiments on the synthetic dataset where we simulated different combinations of these features.

Traffic Game

Traffic game [Chen et al., 2018] is a stateless coordination game motivated by traffic control task where players need to select a route such that they do not cause congestion. $\mathcal{N}$ players present in the domain need to coordinate in a way such that congestion on route k, $l_k = \sum_{i\in\mathcal{N}} 1_{(x_i=k)}$ is neither too many or too few, where x_i is the route selection of agent i. The *reward* function for maintaining desired congestion on a route k is Gaussian function $l_k \cdot e^{-(l_k-\mu_k)^2/\sigma_k^2}$ where μ_k is the desired mean value of congestion and σ_k is the penalty for deviation from the mean value. All the players on the same route receives same reward. The objective of a central traffic controller is to have congestion on each route to be as close to the respective mean value as possible. Hence its goal is to maximize the social welfare whereas the other players are self interested player maximizing their own reward.

When does the meeting start?

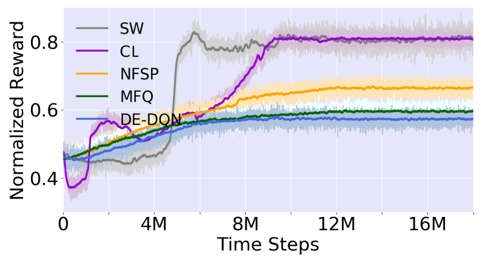
"When does the meeting start?" Guéant et al. [2011] is a stateless coordination game where a number of participants need to attend a meeting. The meeting is scheduled to start at T but it will start only after a minimum number of participants are present for the meeting. Hence very often it starts several minutes after the scheduled time. As a result, each participant i has their own preference T_i when they would like to arrive for the meeting. Also, when participant i decides to arrive at t_i, in reality he arrives at $t_i \pm \sigma_i$ due to uncertainties (traffic etc.). More precisely, t_i is the action taken by the participant and σ_i is the uncertainty he is subjected to.

To decide about their arrival time, each participant optimize their cost. Suppose the meeting started at t and participant i arrived at t_i, then he incurs following cost

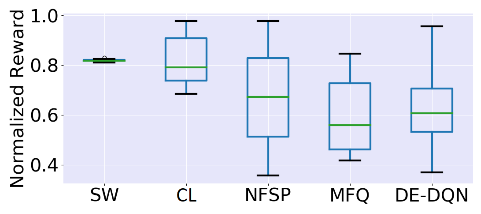
$$\beta_1(|t - t_i|) + \beta_2(T_i - t_i) + \beta_3 max(0, t_i - T)$$

Here first part is lateness/waiting cost, second part is deviation from the preference cost and the last part is reputation cost. β parameters are weigtages of these cost components. Participants learn when to arrive given T and T_i such that their individual cost is minimum. We can assume presence of a central entity (say manager of the team whose member are supposed to attend the meeting) whose objective is to minimize the social welfare cost.

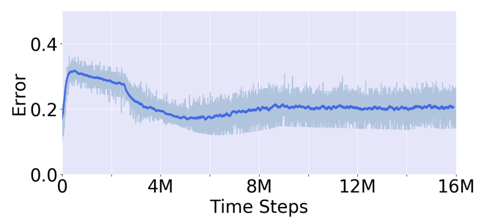
8 RESULTS



(a) Social Welfare



(b) Variance in individual revenue



(c) Difference in central agent's social welfare policy and aggregated policy of individual agents

Figure 2: Taxi simulator

We compare CL with three baseline algorithms: (1) density entropy based deep Q-learning (DE-DQN) [Verma et al., 2019], (2) neural fictitious self play (NFSP) [Heinrich and Silver, 2016] and (3) mean-field Q-learning (MFQ) [Yang et al., 2018]. DE-DQN is an independent learning algorithm which learns from local observation. It predicts agent population density distribution and uses entropy of population density distribution to improve individual learning. NFSP is a fictitious self play learning algorithm where agents learn from their local observation. As CL has advantage of having a central agent providing more information, for fair comparison we provided joint action information to NFSP. We compared with original NFSP as well, but it performed worse than the NFSP with joint action information. Hence we do not include those results here. As discussed in Section 6, we considered the action space of the central agent to be continuous for all our experimental domains.

DE-DQN algorithm is suitable for the setup with partial observation and uses entropy of population density distribution to improve learning. As we consider full view of joint action in case of the two stateless coordination game example, we do not perform experiments for DE-DQN for these examples.

A centralized cooperative learning will ensure an optimal social welfare revenue given the reward function and transition function are same for all the individual agents. Hence, as an upper bound we also compare with social welfare (SW) policy where the all the agents execute the social welfare policy cooperatively.

We evaluated the performance of all learning methods on two key metrics:

- Social welfare payoff computed by aggregating payoffs of all the individual agents. Higher values imply better performance.
- Variation in payoff of individual agents after the learning has converged. This is to understand if agents can learn well irrespective of their initial set up and experiences. This in some ways represents fairness of the learning approach. We use box plots to show the variation in individual revenues, so smaller boxes are better.

Payoff of all the agents is reset after every evaluation period time steps[2]. For taxi simulator and online to offline service simulator one evaluation period consisted of 1000 (1e3), whereas for traffic game and "when does the meeting start" experiments, it was set to 100 time steps. The graphs where social welfare has been compared provide running average of revenue over 100 evaluation periods for taxi simulator and online to offline service simulator, whereas for the remaining two experimental domains it provides running average of 20 evaluation periods.

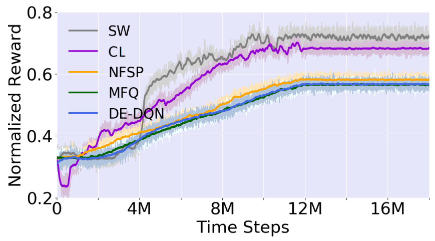
Hyperparameters

Our neural network consisted of one hidden layer with 256 nodes. We also used dropout layer between hidden and output layer to prevent the network from overfitting. dam optimizer for all the experimental domains. We used AFor taxi simulator and online to offline service simulator the learning rate was set to 1e-5 whereas for the remaining two stateless experiments it was set to 1e-4. For all the methods we performed ϵ-greedy exploration and ϵ was decayed exponentially. Training is stopped once ϵ decays to 0.05. In all the experiments, each individual agent maintained a separate neural network. We experimented with different values of aniticipatory parameter for NFSP and used 0.1 which provided the best results.

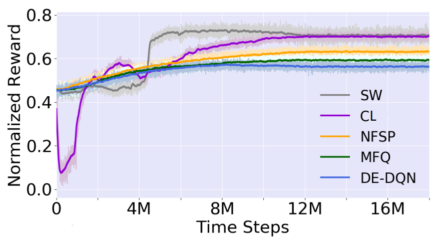
Taxi Simulator

Figure 2 presents the performance comparison for taxi

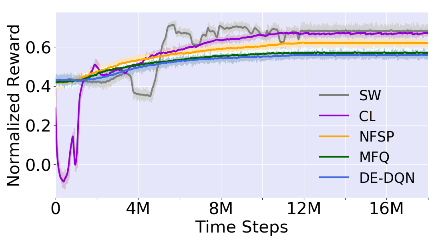
[2]A time step represents a decision and evaluation point in the simulator.



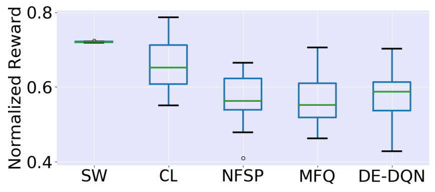
(a) DAR=0.4



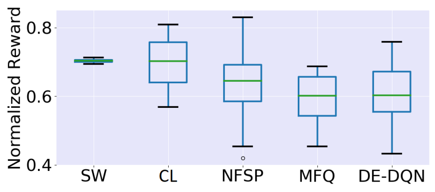
(b) DAR=0.5



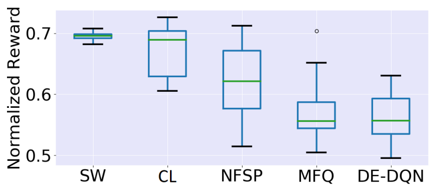
(c) DAR=0.6



(d) DAR=0.4



(e) DAR=0.5



(f) DAR=0.6

Figure 3: Social welfare and variance in individual revenues comparison for online to offline service simulator

simulator where the zone structure and demand distribution were simulated using real-world data. In Figure 2a we can see that CL's social welfare value is similar to that of centralized cooperative learning. However Figure 2b shows that variance in the payoff of individual agents is minimum for SW. This is expected as all the agents follow the same mixed policy computed by the central agent. Furthermore variance for CL is lower than the other three algorithms.

As seen in Figure 2a, the social welfare value of CL is $\approx 13-20\%$ higher than the other three algorithms. It means that there are some "lost demand" (demands that were not served) for NFSP, MFQ and DE-DQN which are being served by SW and CL.

Figure 2c displays error between central agent's social welfare policy and the aggregated joint policy of individual agents. To compute aggregated joint policy of individual agents in zone z, we compute best response of agents present in the zone and then compute $\boldsymbol{a}^z_{bs} = (a^k_{bsz})_{k\in\mathcal{A}_z}$, where a^k_{bsz} is the fraction of agents present in zone z whose best response is to move to zone k. Then we compute root mean squared error (RMSE) between vectors $\boldsymbol{a}^z_c$ and $\boldsymbol{a}^z_{bs}$ to compute error in policies for zone z. Finally, we compute average error over all the zones to get the error between social welfare policy and the joint best response policy.

We can see that the social welfare policy and the joint best response policy converges to different policies as the error between them does not go down to zero. However our argument in Section 5.1 that the policy converges to a stationary policy is validated empirically here.

Online to offline service simulator

For online to offline service simulator, we used multiple combination of agent population size and number of zones (20 agents-10 zones, 20-agents-15 zones, 30 agents-15 zones, 50-agents-25 zones etc.). The trip pattern is uniform and demand arrival rate is static until specified otherwise.

Figure 3 presents plots for social welfare and boxplots for variance in individual revenue for various experimental setups. The first result is for a setup with dynamic arrival rate, non-uniform trip pattern with DAR=0.4. Social welfare value of CL is ≈10-12% more than NFSP, MFQ and DE-DQN (Figure 3a). In Figure 3d we can see that the best agent of CL generates revenue more than its counterpart for the algorithms.

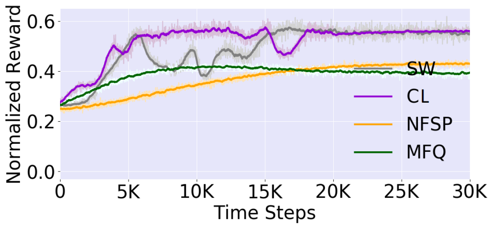
For setup with dynamic arrival rate and DAR=0.5 in Figure 3b, social welfare value of CL is ≈8-12 % more than NFSP, MFQ and DE-DQN. Also as seen in boxplots of Figure 3e, there are around 50 % of the agents who earn more than the social welfare value of SW. We observed similar results for the setup with non-uniform trip pattern and DAR=0.6 (Figures 3c and 3f), CL generated ≈6-10 % more social welfare revenue than NFSP, MFQ and DE-DQN.

Here are the key conclusions:

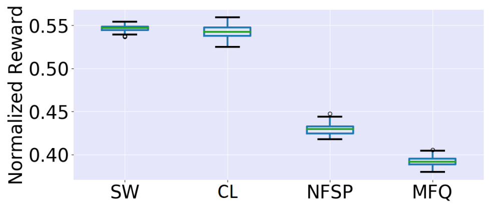
- SW performs best due to its cooperative learning and uniformity of the agents.
- Performances of NFSP, MFQ and DE-DQN with respect to SW vary with varying complexity of the experimental setup (with and without dynamic arrival

rate and non-uniform trips). However CL's social welfare was consistently at par with SW's social welfare. This provides the motivation to the central entity to compute social welfare policy and share it with the non-cooperative individual learners.

- Apart from having lesser variance in the individual revenues, the best and worst performing agents of CL always perform better than the best and worst performing agents of NFSP, MFQ and DE-DQN respectively. This provides motivation for the individual agents to use CL instead of other individual learning algorithms.
- A considerable number (20-50%) of individual agents earn more than the social welfare value of SW. This justifies for the self-interested agents to play best response policy (CL) instead of learning cooperatively (SW).



(a) Social Welfare



(b) Variance in individual reward

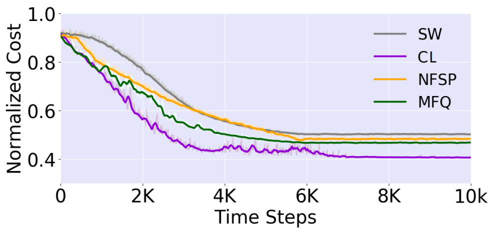
Figure 4: Traffic game

Traffic Game

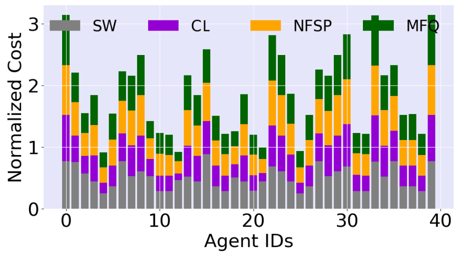
Figure 4 shows results for traffic game with 100 agents with 10 routes. We generated different values and μ and σ for each route to make the learning more complex (as opposed to having same μ and σ for every route). As seen in previous example domains, the social welfare value of CL is similar to that of SW and is ≈15-18% more than NFSP and MFQ. As it is stateless game, other algorithms were also able to achieve low variance in the individual values.

When does the meeting start?

We performed experiments with 40 participants with options of arriving at 15 different time steps. We used normal distribution to introduce uncertainty σ_i and β values were set to 1. Each participant had their own preference



(a) Social Welfare



(b) Individual costs of 40 agents

Figure 5: Social welfare and individual costs for "when does the meeting start?" experiment

of arrival time which makes the domain asymmetric and hence, a central policy might not provide a social optimal. Figure 5a confirms this and we see that performance of SW is worst. Also, cost for CL is ≈6-8 % lesser than MFQ and NFSP. Figure 5b illustrates costs of 40 participants where the results has been shown as stacked bar chart. Each bar represents cost of single participant for these forur algorithms. We can observe that the cost of individual agents are minimum for CL.

9 CONCLUSION

In this work we present correlated learning (CL) for aggregation systems. We exploit the presence of a central entity in aggregation systems to use them as a correlation agent where non-cooperative individual agents learn to play best response against a social welfare policy. We first provide a generic CL and then extend it to use in anonymous domains. Our experiments on multiple example domains show that both central agent and individual agents get benefited by using CL as compared to other individual learning algorithms.

Acknowledgements

This research is supported by the National Research Foundation Singapore under its Corp Lab @ University scheme and Fujitsu Limited as part of the A*STAR-Fujitsu-SMU Urban Computing and Engineering Centre of Excellence.

References

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 17, pages 1021–1026.

Brown, G. W. (1951). Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13:374–376.

Carmona, R. and Delarue, F. (2014). The master equation for large population equilibriums. In *Stochastic analysis and applications*, pages 77–128.

Chen, Y., Zhou, M., Wen, Y., Yang, Y., Su, Y., Zhang, W., Zhang, D., Wang, J., and Liu, H. (2018). Factorized q-learning for large-scale multi-agent systems. *arXiv preprint arXiv:1809.03738*.

Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2018). Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 122–130.

Greenwald, A., Hall, K., and Serrano, R. (2003). Correlated q-learning. In *International Conference on Machine Learning (ICML)*, volume 3, pages 242–249.

Guéant, O., Lasry, J.-M., and Lions, P.-L. (2011). Mean field games and applications. In *Paris-Princeton lectures on mathematical finance 2010*, pages 205–266.

Harsanyi, J. C., Selten, R., et al. (1988). A general theory of equilibrium selection in games. *MIT Press Books*, 1.

Heinrich, J., Lanctot, M., and Silver, D. (2015). Fictitious self-play in extensive-form games. In *International Conference on Machine Learning (ICML)*, pages 805–813.

Heinrich, J. and Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research (JMLR)*, 4(Nov):1039–1069.

Hu, J., Wellman, M. P., et al. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *International Conference on Machine Learning (ICML)*, volume 98, pages 242–250.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Perolat, J., Silver, D., Graepel, T., et al. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4190–4203.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163.

Littman, M. L. (2001). Friend-or-foe q-learning in general-sum games. In *International Conference on Machine Learning (ICML)*, volume 1, pages 322–328.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6379–6390.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529.

Nguyen, D. T., Kumar, A., and Lau, H. C. (2017). Policy gradient with value function approximation for collective multiagent planning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4319–4329.

Nutz, M. (2018). A mean field game of optimal stopping. *SIAM Journal on Control and Optimization*, 56(2):1206–1221.

Sandholm, W. H. (2010). *Population games and evolutionary dynamics*. MIT press.

Shoham, Y., Powers, R., and Grenager, T. (2003). Multi-agent reinforcement learning: a critical survey. *Web manuscript*.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Verma, T., Varakantham, P., and Lau, H. C. (2019). Entropy based independent learning in anonymous multi-agent settings. *International Conference on Automated Planning and Scheduling (ICAPS)*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Weinberg, M. and Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 506–513.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. (2018). Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 5567–5576.