Optimal Swarm RL: An improved Deep Exploration strategy

Rishav Chourasia
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati
Guwahati, Assam 781039
r.chourasia@iitg.ac.in

Abstract

In this paper, we define a class of Ensemble reinforcement learning (RL) algorithms, which we call *Swarm RL*. Presenting an analysis on policy improvement property, convergence and parameterized regret bounds of Swarm RL, we develop an algorithm with regret-optimal action selection strategy, called *Optimal Swarm RL*. Interestingly, many other notable RL algorithms like the Bootstrapped Q-learning(Q-learning version of Bootstrapped DQN), Q-Ensemble voting also belong to Swarm RL and thus Optimal Swarm RL dominates both of these, as shown by simulations on two MDPs designed to test deep exploration. The core idea behind the Optimal Swarm RL decision making policy is to optimally cluster agents based on action decision for every observation. We show that determining this clustering reduces to the optimal Set Cover problem on input size of ensemble cardinality times action set size. Optimal Swarm RL results from a direct consequence of minimizing the parameterized regret bound, which decreases exponentially as the clusters reach maximal sizes. The optimal clustering improves communication between ensemble heads, therefore facilitating deep exploration.

1 Introduction

Many problems, when modeled as a Reinforcement Learning (RL) environment, have a sparse distribution of rewards leading to naive exploration strategies, such as Boltzmann or ϵ-greedy, performing inefficiently due to an inability of planning over multiple state sequences[1]. Other exploration strategies like Bayesian Q-learning[2], that maintains uncertainty over expected net returns via probability distribution, or PSRL[3], that maintains distribution over MDPs, is intractable for practical problems. Using Deep Q-Learning Network(DQN) ensembles to maintain a sample over expected net returns instead was adopted to tackle this intractability, which led to the problem of achieving efficient shared learning between ensemble heads[4, 5]. Various strategies like Ensemble Voting DQN[6], Averaged DQN[7], Bootstrapped DQN[8] were proposed to achieve such a sharing. While Bootstrapped DQN tried to share learning via Experience Replay Buffer, Ensemble Voting DQN suggested sound action decision with majority consensus. Averaged DQN proposed sharing learning via averaging of ensemble head estimates for error calculation. Efficacy of knowledge sharing has a direct impact on the efficiency of the algorithm, and so it is worthwhile to try and develop algorithms with better knowledge sharing mechanisms.

Based on method for error calculation and action decision for training an ensemble using heads' estimates, a class of algorithms, we call Swarm RL, can be formulated with guaranteed convergence as well as preserved optimality, provided the update strategy abides by an improvement property. Ensemble algorithms such as Q-Ensemble Voting and Bootstrapped Q-learning belong in this class of RL algorithms. A Swarm RL algorithm in general employs sharing of knowledge via action decisions

Preprint. Work in progress.

for achieving deep exploration. In another words, this class of algorithm has available an implicit channel of communication via knowledge about action preferences, which can be utilized for more efficient learning.

Among Swarm RL algorithms, we are interested in regret-optimal Swarm RL algorithm, the one that achieves the best communication and consequently has the least expected regret overall. By developing regret bounds on an arbitrary Swarm RL algorithm, important insights on what constitutes as the best algorithm in this class can be developed. Moreover, it might help in determining the number of ensemble heads to employ for best performance guarantees.

Our intuition for the Optimal Swarm RL algorithm is that an effective communication between heads could be established by grouping similar agents, and training them using in-sync action decisions while error computation. These groupings should be different with each observed state and should depend on the agents' net returns estimates. We draw parallels with an observation in real-life group decision making that proponents of similar decisions usually collude together to form as big a party as possible and take a uniform decision. Multiple parties promoting mostly disjoint decisions come to exist and if smaller parties have fairly coherent ideologies, they form coalitions. The analysis of the regret bound, as we will see, formalizes this intuition in form of an action selection strategy which selects set of coherent actions, which surprisingly obeys an extreme form of Justified Representation: a desirable property in approval-based committee voting.

After a brief background on notation, Set Cover problem and some ensemble DQN algorithms, we introduce a weak policy improvement theorem and adapt it to formulate a class of ensemble reinforcement learning algorithms, we call Swarm RL. Q-Ensemble Voting and Bootstrapped Q learning are shown to belong in this class. Furthermore, we show that every algorithm in this class eventually converges, giving the optimal policy. After this, we develop a bound on expected ensemble-regret of an arbitrary Swarm RL and analyzing this bound, we develop a clustering mechanism which leads to the regret-optimal Swarm RL algorithm. This clustering mechanism reduces to the optimal Set Cover problem. Employing an approximate Greedy Set Cover algorithm, we present a DQN adaptation we call Optimal Swarm DQN and compare its results on MDP toy problem developed by Osband et al. [8] for depicting deep exploration of Bootstrapped DQN.

2 Background

2.1 Notation

We consider a typical RL environment as a Markov Decision Process (MDP) $M := (S, A, T, R, \gamma)$, where S is the state space, A is the action space and γ is the discount factor belonging in the range $[0, 1)$. For any $s, s' \in S$ and $a \in A$, $R(s, a)$ gives the finite reward of taking action a at state s and $T(s'|s, a)$ gives the environment's transition probability of going to state s' on taking action a at state s. Without any loss in generality, we assume that the range of the bounded reward function R is $[0, 1 - \gamma]$ to make the maximum discounted returns independent of γ. Any other bounded reward function can always be linearly scaled down to this range without affecting the action decisions. We also denote the space for all bounded functions with range $[0, 1]$ on $S \times A$ and on S as $\mathcal{Q} := \mathcal{Q}_{S,A}$ and $\mathcal{V} := \mathcal{V}_S$ respectively. This bound on range reflects the bound on net returns any policy can get when facing R reward function.

A mapping $\pi : S \to A$ is referred to as a deterministic policy. Every policy π has a corresponding Q function $Q^\pi \in \mathcal{Q}$ computed from the Bellman equation

$$Q^\pi(s, a) = R(s, a) + \gamma E_{s' \sim T(\cdot|s,a)}[Q^\pi(s', \pi(s'))]. \tag{1}$$

From now on, we'll represent $E_{s' \sim T(\cdot|s,a)}$ as E_T for convenience. This $Q^\pi(s, a)$ function is interpreted as the expected discounted net returns an agent receives on starting with action a on state s and following policy π thereafter. The value function

$$V^\pi(s) = Q^\pi(s, \pi(s)) \tag{2}$$

can similarly be interpreted as the discounted net returns on following policy π right from the first state s. An optimal policy π^* is defined as a policy mapping that accrues maximum expected discounted returns on MDP M, whose Q-table Q^* was shown by Tsitsiklis and Bertsekas [9] to be the unique identity solution of the Bellman operator $\mathcal{T}|\mathcal{Q} \to \mathcal{Q}$, defined point-wise as

$$\mathcal{T}Q(s, a) = R(s, a) + \gamma E_T[\max_b Q(s', b)]. \tag{3}$$

Starting from any Q-function $Q_0 \in \mathcal{Q}$, a sequence of application of $\mathcal{T}$ leads to convergence to the optimal Q-function Q^*[10] i.e.

$$Q^*(s,a) = \lim_{t\to\infty} \mathcal{T}^t Q_0(s,a). \tag{4}$$

and corresponding value function i.e.

$$V^*(s) = \max_a Q^*(s,a). \tag{5}$$

Dealing with an ensemble of K agents, we denote the initial Q-functions as $Q_0^{(i)} \in \mathcal{Q} \quad \forall i \in \mathcal{K} := \{1, 2..K\}$. Let $H_t = (s_0, a_0, R(s_0, a_0), s_1, a_1, ..., s_{t-1}, a_{t-1}, R(s_{t-1}, a_{t-1}))$ denote the history of observations made before t^{th} update step. Depending on the RL algorithm, say represented by the operator $\mathcal{T}'$, the returns estimates $Q_t^{(i)}, \forall i \in \mathcal{K}$ at t^{th} step for a history, H_t, could be said to have been sampled from the posterior distribution over $\mathcal{Q}$ [3, 11], i.e.

$$Q_t^{(i)} | H_t \sim P_{\mathcal{T}'}(Q|H_t), \quad \forall i \in \mathcal{K}. \tag{6}$$

For a distribution on MDPs, respective Q^* and V^* are stochastic and follow some distribution depending upon M's distribution. For such a stochastic environment, we denote ensemble regret $Regret_{\pi^{(\cdot)}}$ for an ensemble policy $\pi^{(\cdot)}$ as

$$Regret_{\pi^{(\cdot)}}(s) = \sum_{i\in\mathcal{K}} \sum_{a\in A} \mathbb{1}\{\pi^{(i)}(s) = a\} \Delta_{s,a}, \tag{7}$$

where $\pi^{(i)}$ is the policy for agent i and $\Delta_{s,a}$ is the expected action regret given by

$$\Delta_{s,a} = E[V^*(s) - Q^*(s,a)]. \tag{8}$$

Range of $\Delta_{s,a}$ is $[-1, 1]$ because of range of any $Q \in \mathcal{Q}$ is $[0, 1]$.

2.2 Set Cover

Given a collection $\mathcal{S}$ of subsets of a universe set $\mathcal{U}$, a subset $C \subset \mathcal{S}$ that covers all the elements of the universe $\mathcal{U}$ is called a Set Cover on the collection $\mathcal{S}$. Mathematically, a valid cover is expressed as

$$\mathcal{U} = \bigcup_{X\in C} X, \quad C \subset \mathcal{S}. \tag{9}$$

An optimal Set Cover is defined as a cover C^* such that for any other cover C, $|C^*| <= |C|$. The problem of finding optimal Set Cover has been shown to be in NP-complete[12]. However, many polynomial time approximate algorithms exists that can be used to find a close to optimal cover.

In our algorithm presented later, the task of clustering of agents at each time step essentially reduces to finding optimal Set Cover for universe $\mathcal{U} = \mathcal{K}$, and collection $\mathcal{S}$ consisting of a set of agents for all action $a \in A$, i.e. $|\mathcal{S}| = |A|$. Though for our examples, the sizes are conveniently small to use exponential time solutions to the Set Cover problem, efficient implementation of polynomial time $log(|\mathcal{U}|)$-approximate Set Cover algorithm[13], adapted from Greedy Set Cover can be used.

2.3 Ensemble DQN algorithms

Ensemble DQN algorithms involve a set of Q-function estimates $Q_t^{(i)}$ parameterized in form of a neural network architecture, usually with some shared initial layers for efficient learning[8]. A shared experience buffer is used for all the heads in which experience tuples of the form $(s_t, a_t, R(s_t, a_t), s_{t+1})$, that are generated by the ensemble, are stored. Ensemble DQN algorithms differ in the action decision mechanism, loss calculation or the way experiences are utilized for training.

For instance, in Bootstrapped DQN, the action decision is the greedy action according to a randomly sampled head's estimate, the head being sampled at the start of each episode. Also, along with the experience tuple, a probability p Bernoulli mask vector $M = (m^{(1)}, m^{(2)}.., m^{(|\mathcal{K}|)})$ is stored, which determines the subset of agents that has access to the tuple for training. Frequently enough, batches of experience tuples are sampled from the replay and used to train the network optimizing for the following loss

$$L = E[((Y_t^{(\cdot)} - Q_t^{(\cdot)}(s,a)) \cdot M)^2], \tag{10}$$

where

$$Y_t^{(\cdot)} = R(s,a) + \gamma \times Q_t^{(\cdot)}(s', \pi_t^{(\cdot)}(s')), \tag{11}$$

and $(s, a, R(s,a), s', M)$ is sampled from experience buffer. The next state action $\pi_t^{(\cdot)}(s')$ is selected greedily as

$$\pi_t^{(i)}(s') = \underset{b}{\text{argmax}}\, Q_t^{(i)}(s', b), \quad i \in \mathcal{K}. \tag{12}$$

Another instance is Ensemble Voting DQN where the action selection is via majority vote, and loss calculation is same as bootstrapped DQN except the masking flag M. Similar to Double DQN, ensemble DQN algorithms can be stabilized in terms of overestimation by using target network's Q-function estimate in $Y_t^{(\cdot)}$.

3 Swarm RL class of ensemble algorithms

The idea behind swarm update strategy comes from a simple observation that given a policy and it's corresponding Q-function, weaker policy updates compared to Bellman's greedy arg-max approach[14] can still lead to an improvement in policy, as we state in the following theorem.

Theorem 3.1 (Weak Policy Improvement Theorem)**.** *If π is a policy mapping states to actions with corresponding converged Q-function Q^π resulting from following the policy deterministically, then for a set of update policies Ω defined such that for any $\omega \in \Omega$,*

$$Q^\pi(s, \omega(s)) \geq Q^\pi(s, \pi(s)), \quad \forall s \in S, \tag{13}$$

has the property that

$$Q^\omega(s,a) \geq Q^\pi(s,a), \quad \forall s \in S, a \in A, \omega \in \Omega. \tag{14}$$

This freedom in selection of update policy allows us an opportunity to improve knowledge sharing among an ensemble of agents by selecting update policies for ensemble such that some form of coherence is maximized among the agents. Note here that the above condition does not uniquely determine ω, but rather defines a class of possible ω that would result in policy improvement.

Similar to the manner in which Value Iteration algorithm was adapted to Q-learning using the idea of Temporal Difference bootstrap[15] to speed up learning[16], this lenient update strategy can also be formalized in form of an Operator for ensembles, quite similar to the Bellman Operator. Continuing from the notation introduced in 2.1, for an ensemble, $\mathcal{K}$, we define Swarm operator $\mathcal{T}_t^{(i)}$ and the corresponding Q-function update at time step t for an agent $i \in \mathcal{K}$ as

$$Q_{t+1}^{(i)}(s,a) = \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) = R(s,a) + \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s'))], \tag{15}$$

where the $\pi_t^{(i)}$ is analogous to ω in sense that it is a weak improvement(rather than greedy arg-max) over last iteration's policy, i.e.

$$\begin{aligned} Q_t^{(i)}(s', \pi_t^{(i)}(s')) &\geq Q_t^{(i)}(s', g_{t-1}^{(i)}(s')), \quad \forall t > 0, \\ \pi_0^{(i)}(s') &= g_0^{(i)}(s'), \end{aligned} \tag{16}$$

$g_{t-1}^{(i)}$ being the greedy strategy for $t-1$ step Q-function,

$$g_{t-1}^{(i)}(s') = \underset{b}{\text{argmax}}\, Q_{t-1}^{(i)}(s', b). \tag{17}$$

For each agent, the policy $\pi_t^{(i)}$ is free to choose between a set of preferable actions,

$$\mathcal{A}_t^{(i)}(s) = \{a | Q_t^{(i)}(s,a) \geq Q_t^{(i)}(s, g_{t-1}^{(i)}(s))\}, \tag{18}$$

as visualized in Figure 1, and we want to exploit this freedom for achieving knowledge sharing via establishment of some sort of coherence. Note that the freedom in the policy selection leads to a class of algorithms, all of which are described by the same sequence of Swarm Operators $\mathcal{T}_t^{(i)}$, but differ by the exact action selection policy $\pi_t^{(i)}$. We refer to this class as *Swarm RL*.

Social Choice theorists might be tempted to imagine preferable action sets (18) of the agents as their Ballot profiles on a set of candidates, A. A committee needs to be formed consisting of a subset $A_t(s) \subset A$ of candidates that satisfies a strict Justified Representation[17] condition that every agent is represented by a preferred candidate in the elected committee, i.e.

$$A_t(s) \cap \mathcal{A}_t^{(i)}(s) \neq \emptyset, \quad \forall i \in \mathcal{K}. \tag{19}$$

Consequently, agent's action decisions can be imagined as one of preferred candidates that got elected, that is

$$\pi_t^{(i)}(s) \in A_t(s) \cap \mathcal{A}_t^{(i)}(s), \quad \forall i \in \mathcal{K}. \tag{20}$$

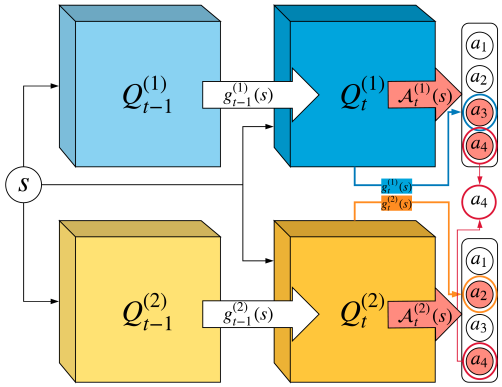

Figure 1: Depicting sets of preferable actions coloured light red, greedy arg-max actions using blue/yellow coloured circles and a coherent action using red coloured circle.

Having formalized the Swarm RL class of algorithms, it is necessary to analyze the convergence and optimality-preserving properties to ensure possibility of any use of these algorithms[18]. Every algorithm in Swarm RL class eventually converges giving the optimal Q-function Q^*, as we state in the following theorem.

Theorem 3.2. *Swarm RL algorithms are optimality preserving, i.e. for any random initialization $Q_0^i \in \mathcal{Q}$ and subsequent application of swarm operators $\mathcal{T}_t^{(i)}$, the asymptotic Q-functions $\widetilde{Q}^{(i)}, \forall i \in \mathcal{K}$ is same as optimal Q-function Q^*, i.e. for all $s \in S, a \in A$,*

$$Q^*(s,a) = \widetilde{Q}^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)} \mathcal{T}_{t-1}^{(i)} \ldots \mathcal{T}_0^{(i)} Q_0^{(i)}(s,a). \tag{21}$$

We point out here that using the operator definition of Swarm RL class (15), one can show that Q-Ensemble Voting algorithm and Bootstrapped Q-learning fall under it.

4 Regret bound and Optimal Swarm RL

Among all Swarm RL algorithms, we are interested in the regret-optimal Swarm RL algorithm. By trying to upper-bound expected ensemble-regret (7) for Swarm RL, we hope to derive an optimal update policy $\pi_t^{(i)}$ that leads to best learning performance. We would observe from the following regret bound on expected returns that the best update policy has an obvious interpretation: agents share *wisdom* in form of their preferred actions and decide upon actions in a manner that dissonance in form of number of distinct action decisions is minimized.

From the definition of ensemble-regret, parameterized on the action recommended by $\pi_t^{(\cdot)}$ for a state s, agglomerations of agents based on unique actions can be created. We add on to the notations defined in 2.1 by adding

$$\mathcal{J}_t^{(a)}(s) = \{i | \pi_t^{(i)}(s) = a\}, \quad \forall a \in A, \tag{22}$$

which represents these agglomerations. In the following theorem, we state the bound on policy parameterized expected ensemble-regret.

Theorem 4.1. *For a swarm RL algorithm, expectation of ensemble regret* $Regret_{\pi_t^{(\cdot)}}(s)$ *at step* t *for observed state* s *is upper-bounded as*

$$E[Regret_{\pi_t^{(\cdot)}}(s)] \leq \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \frac{|\mathcal{J}|\Delta_{s,a}}{e^{\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}}. \tag{23}$$

From the above regret bound inequality, one can see that the bound is minimized when the number of agglomerations, $|\mathcal{J}_t^{(\cdot)}(s)|$, is minimized and individual agglomeration sizes, $|\mathcal{J}_t^{(i)}(s)|$, are maximized. In other words, our swarm optimal regret policy, $\pi_t^{(\cdot)}(s)$, must select actions from corresponding preferable action sets, $\mathcal{A}_t^{(\cdot)}(s)$, such that number of distinct actions are minimized.

This is in alignment with our intuition that an optimal algorithm would employ a mechanism for selecting actions for which largest possible parties with mutually disjoint action decisions are formed. The formulation of parties are dependent on the individual action decisions, which in turn depends on the shared wisdom of agents' preferred action sets.

The restriction imposed by (16) coupled with the above observation induces the requirement of selecting a minimal action subset $A_t(s) \subset A$ such that

$$\bigcup_{a \in A_t(s)} \mathcal{I}_t^{(a)}(s) = \mathcal{K}, \tag{24}$$

where $\mathcal{I}_t^{(\cdot)}(s)$ is the set inverse of $\mathcal{A}_t^{(\cdot)}(s)$ that maps actions to sets of agents that find the action preferable and is defined as

$$\mathcal{I}_t^{(a)}(s) = \{j | a \in \mathcal{A}_t^{(j)}(s)\}. \tag{25}$$

The resultant optimal policy for an agent i must select an action that belongs to the intersection of set of personally preferred actions $\mathcal{A}_t^{(i)}(s)$ and the set of coherent group decision actions $A_t(s)$, as shown in (20)

In simple terms, forming the smallest possible committee of candidates $A_t(s)$ so that the strict Justified Representation condition (19) is satisfied and selecting the corresponding policy using (20) leads to the tightest regret bound. This mechanism of action decision forms the core of Optimal Swarm RL algorithm.

As mentioned in section 2.2 and from analyzing (24) and (25), $A_t(s)$ can also be interpreted as the optimal Set Cover for sets $\mathcal{I}_t^{(\cdot)}(s)$ on a universe $\mathcal{K}$.

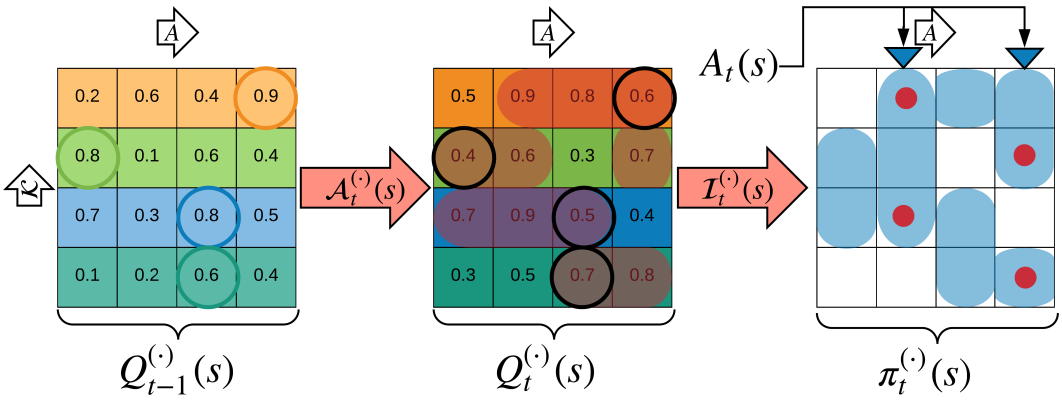

Figure 2: Example of optimal policy selection for state s at time step t for an ensemble of four agents

Figure 2 gives an example of this optimal policy selection mechanism. Rows of the first and second matrices depict agent's Q-function values for time step $t-1$ and t respectively. Coloured circles in first matrix represent greedy actions, $g_{t-1}^{(\cdot)}(s)$, which become thresholding actions in second matrix. Set $\mathcal{A}_t^{(\cdot)}(s)$ is depicted by transparent rounded rectangles in second matrix, which is then inverted to get $\mathcal{I}_t^{(\cdot)}(s)$ in the third matrix. Blue triangles represent the final cover $A_t(s)$, and the red dots are possible regret optimal action choices $\pi_t^{(\cdot)}(s)$.

5 Optimal Swarm DQN algorithm

Using the idea of Deep Q-learning Networks[19], we can adapt the Optimal Swarm RL algorithm into Optimal Swarm DQN as presented in Algorithm 1. Optimal Swarm RL requires maintaining Q-function estimate of the previous iteration. Instead of doing the same for the DQN adaptation, we reuse the target network, which was employed for stabilizing learning[19] as well as removing optimistic overestimation[20, 21], for also finding sets of preferred actions. For every episode, just like in Bootstrapped DQN, a head is randomly sampled and the Optimal Swarm action decision for the head's cluster is followed. Sampling head per observation disrupts deep exploration[2, 22], and therefore isn't done.

Algorithm 1 Optimal Swarm DQN algorithm

procedure AGGLOMERATE(state s, Target Network $T^{(\cdot)}$, Current Network $Q^{(\cdot)}$)
  Find Target best actions $g^{(i)} = \text{argmax}_{a \in A}\, T^{(i)}(s, a)$
  Construct preferred action sets $\mathcal{A}^{(i)} = \{a | Q^{(i)}(s, a) \geq Q^{(i)}(s, g^{(i)})\}$
  Revert preferred actions sets to get $\mathcal{I}^{(a)} = \{i | a \in \mathcal{A}^{(i)}\}$
  Get optimal Set Cover of $\mathcal{K}$ from $\mathcal{I}^{(\cdot)}$. Delete sets in $\mathcal{I}^{(\cdot)}$ that were not in optimal cover
  return $\mathcal{I}^{(\cdot)}$
Initialize $\mathcal{K}$ current $Q_0^{(i)}$ and target $T^{(i)}$ heads $T^{(i)} = Q_0^{(i)}$ with random weights.
Initialize step $t \leftarrow 0$
for Each episode **do**
  Pick an agent at random using $i \sim Uniform(\mathcal{K})$
  for Each step observation s_t of episode **do**
    $\mathcal{J}^{(\cdot)} \leftarrow$ Agglomerate(s, $T^{(\cdot)}$,$Q_t^{(\cdot)}$)
    Take action $a_t = a$, s.t. $i \in \mathcal{J}^{(a)}$, and observe r_t and s_{t+1}.
    Store experience (s_t, a_t, r_t, s_{t+1}) in replay memory D.
    for $(s_{t'}, a_{t'}, r_{t'}, s_{t'+1})$ in *SampleBatch(D)* **do**
      $\mathcal{J}^{(\cdot)} \leftarrow$ Agglomerate(s, $T^{(\cdot)}$,$Q_t^{(\cdot)}$)
      Compute $a_{t'+1}^{(j)} = a$, s.t. $j \in \mathcal{J}^{(a)}, \forall\, j \in \mathcal{K}$
      Train using error = $E^{(\cdot)} \leftarrow r_{t'} + \gamma \times T^{(\cdot)}(s_{t'+1}, a_{t'+1}^{(\cdot)}) - Q_t^{(\cdot)}(s_{t'}, a_{t'})$
    if t % *UpdateFreq* = 0 **then** $T^{(\cdot)} \leftarrow Q_t^{(\cdot)}$
    $t \leftarrow t + 1$.

The *Agglomerate*$(s, T^{(\cdot)}, Q^{(\cdot)})$ procedure is used while making action decisions as well as for error calculation during training on a sampled batch. This procedure uses exponential time Optimal Set Cover algorithm for generating a cover of $\mathcal{K}$, which isn't computationally restrictive as long as the number of heads in the ensemble and action set size are small. As increasing either of them directly increases neural network architecture complexity, usually they are kept small. For large number of heads, $log(|\mathcal{K}|)$-approximate greedy algorithm for optimal Set Cover can be used.

6 Experiments and results

We performed comparisons of Optimal Swarm DQN algorithm with Ensemble Voting DQN and Bootstrapped DQN on a linear MDP toy problem (taken form [8]) with $n = 15$ and it's two dimensional, more difficult grid extension with $n = 9 \times 9$. The toy problems are designed to test RL algorithms' ability for deep exploration, as starting at state s_2 (state $s_{2,2}$ in grid MDP), probability of randomly visiting the closest, lesser rewarding terminal state s_1 (state $s_{1,1}$ for grid MDP) is exponentially greater than that of the higher rewarding terminal state s_{15} (state $s_{9,9}$ in grid MDP). As seen from figure 4, Optimal Swarm DQN shows a much faster convergence for both the problems, indicating an anticipated improvement in exploration.

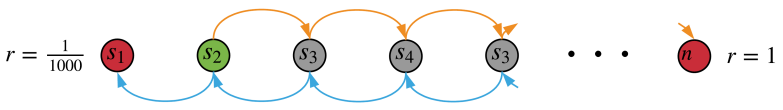

Figure 3: Linear MDP toy problem

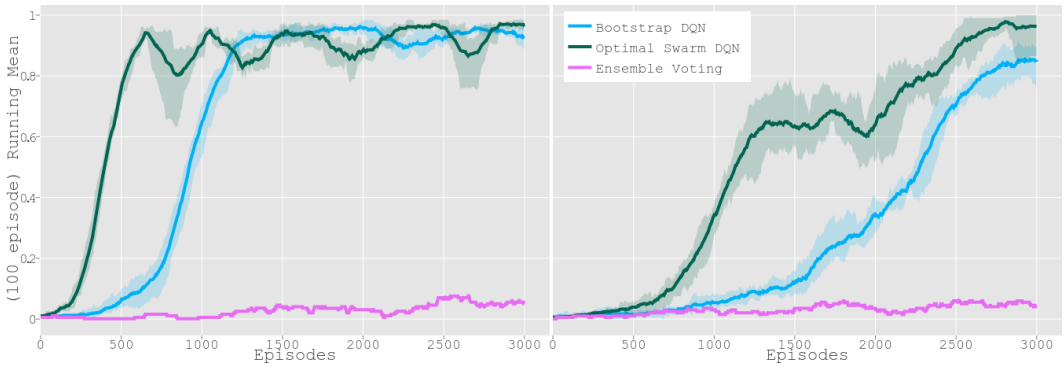

Figure 4: Comparison of ensemble RL algorithms on linear MDP (left) and grid MDP (right)

Osband et al. [8] observed that in Bootstrapped DQN, randomly masking experience tuples for heads had a rather detrimental affect, which indicates a possible bottleneck in the rate of information exchange. Since in Bootstrapped DQN, the only means of communication in between the heads is via unmasked tuples in the experience replay buffer, masking them could be throttling the information exchange occurrence leading to decline in performance.

(a) Avg. episodic returns for Bootstrapped DQN (b) Avg. episodic returns for Optimal Swarm DQN

Figure 5: Avg returns for (a)Bootstrapped DQN's and (b)Optimal Swarm DQN's ensemble heads

The comparison of ensemble heads' growth in episodic returns for Bootstrapped and Optimal Swarm DQN for linear MDP is shown in figure 5. One can notice that the returns plot for Bootstrapped DQN's heads are relatively disparate, noisy and converges significantly slower than Optimal Swarm DQN, confirming that a communication bottleneck is indeed happening for Bootstrapped DQN and is mitigated, if not removed entirely, for Optimal Swarm DQN.

7 Concluding remarks

We presented in this paper, a family of Ensemble RL algorithms that are asymptotically convergent and optimality preserving. Q-Ensemble Voting and Bootstrapped Q-learning are two distinguished members of this class. For an arbitrary algorithm in this class, we developed a policy parameterized regret-bound, using which we developed the regret-optimal action decision policy for this class. On drawing parallels between an arbitrary action decision mechanism and approval-based committee voting problem, equivalence between smallest committee satisfying strict Justified Representation and regret-optimal action decisions emerged as an interesting observation. We showed that finding this decision mechanism reduces to optimal Set Cover problem of usually small input sizes. A scalable and tractable adaptation of the corresponding algorithm called Swarm DQN was presented and for the proof of concept, an improved exploration and learning rate was shown compared to Bootstrapped DQN and Ensemble Voting DQN on linear and grid MDPs. We believe that our approach gracefully generalizes classical Q-learning over ensembles and provides an improved deep exploration strategy.

Acknowledgments

We thank Nikhil Yadala, Research Engineer at Microsoft, for sharing his wisdom during the course of research and Rishabh Jangir, Graduate Researcher at Institut de Robòtica i Informàtica Industrial, for comments that greatly improved the manuscript.

References

[1] Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems 26*, pages 3021–3029. 2013.

[2] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. pages 761–768, 1998.

[3] I. Osband, B. Van Roy, and D. Russo. (more) efficient reinforcement learning via posterior sampling. 2013.

[4] Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *AAAI*, pages 1640–1646, 2017.

[5] R. R Menon and B. Ravindran. Shared Learning : Enhancing Reinforcement in Q-Ensembles. *arXiv preprints arXiv:1709.04909*, 2017.

[6] M. A. Wiering and H. van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

[7] O. Anschel, N. Baram, and N. Shimkin. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. *arXiv preprint arXiv:1611.01929*, 2016.

[8] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems 29*, pages 4026–4034. 2016.

[9] J Tsitsiklis and D Bertsekas. Neuro-dynamic programming. *Athena Scientific*, 1996.

[10] Takashi Kamihigashi et al. Existence and uniqueness of a fixed point for the bellman operator in deterministic dynamic programming. Technical report, 2012.

[11] S. Agrawal and R. Jia. Posterior sampling for reinforcement learning: worst-case regret bounds. *arXiv preprint arXiv:1705.07041*, 2017.

[12] Dorit S Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143, 1996.

[13] Tapio Elomaa and Jussi Kujala. Covering analysis of the greedy algorithm for partial cover. In *Algorithms and Applications*, pages 102–113, 2010.

[14] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[15] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1): 9–44, 1988.

[16] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[17] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh. Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485, 2017.

[18] Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI*, pages 1476–1483, 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[20] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[21] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. pages 2094–2100, 2016.

[22] Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2012.

APPENDICES

A Theoretical results

Theorem A.1 (Weak Policy Improvement Theorem)**.** *If π is a policy mapping states to actions with corresponding converged Q-function Q^π resulting from following the policy deterministically, then for a set of update policies Ω defined such that for any $\omega \in \Omega$,*

$$Q^\pi(s, \omega(s)) \geq Q^\pi(s, \pi(s)), \quad \forall s \in S, \tag{26}$$

has the property that

$$Q^\omega(s, a) \geq Q^\pi(s, a), \quad \forall s \in S, a \in A, \omega \in \Omega. \tag{27}$$

Proof. We will prove this using induction on the horizon depth h. For every h, we have a corresponding Q-function for following policy π deterministically on each step of the horizon. This is given by the equation

$$\begin{aligned} Q_h^\pi(s, a) &= R(s, a) + \gamma E_T[Q_{h-1}^\pi(s', \pi(s'))], \\ Q_1^\pi(s, a) &= R(s, a). \end{aligned} \tag{28}$$

where $s' \sim T(\cdot|s, a)$. We represent the depth-h arbitrary update policy ω_h defined to obey the constraint

$$Q_h^\pi(s, \omega_h(s)) \geq Q_h^\pi(s, \pi(s)), \quad \forall s \in S. \tag{29}$$

The Q-function for following the update policy ω_h on corresponding horizon depth h is represented as $Q_h^{\omega_h}$ and given by the equation

$$\begin{aligned} Q_h^{\omega_h}(s, a) &= R(s, a) + \gamma E_T[Q_{h-1}^{\omega_{h-1}}(s', \omega_{h-1}(s'))], \\ Q_1^{\omega_1}(s, a) &= R(s, a). \end{aligned} \tag{30}$$

By subtracting (28) from (30), we get

$$Q_h^{\omega_h}(s, a) - Q_h^\pi(s, a) = \gamma E_T[Q_{h-1}^{\omega_{h-1}}(s', \omega_{h-1}(s')) - Q_{h-1}^\pi(s', \pi(s'))]. \tag{31}$$

Induction Hypothesis: For all horizon depth h, $Q_h^{\omega_h}(s, a) \geq Q_h^\pi(s, a), \forall s \in S, a \in A$.

Limiting step: For $h = 1$ case, it is clear that $Q_1^{\omega_1}(s, a) \geq Q_1^\pi(s, a)$, as both are equal to $R(s, a)$.

Inductive step: Assuming the induction inequality is true for depth $h - 1$, we need to show that it is also true for depth h. From (31), the inequality for depth h is true if

$$Q_{h-1}^{\omega_{h-1}}(s', \omega_{h-1}(s')) - Q_{h-1}^\pi(s', \pi(s')) \geq 0, \quad \forall s' \in S. \tag{32}$$

Consider two cases.

Case 1. *If $\omega_{h-1}(s') = \pi(s') = a'$(say), then the above inequality follows from the induction hypothesis on $h - 1$.*

Case 2. *If $\omega_{h-1}(s') \neq \pi(s')$, then we have*

$$Q_{h-1}^{\omega_{h-1}}(s', \omega_{h-1}(s')) \geq Q_{h-1}^\pi(s', \omega_{h-1}(s')), \quad \forall s' \in S, \tag{33}$$

from the induction hypothesis and

$$Q_{h-1}^\pi(s', \omega_{h-1}(s')) \geq Q_{h-1}^\pi(s', \pi(s')), \quad \forall s' \in S, \tag{34}$$

from the definition update policy ω_{h-1} for depth $h - 1$. Combining both (33) *and* (34) *we get the required inequity* (32)*.*

Thus the induction hypothesis is true for all depth h. On applying $\lim h \to \infty$ to both the sides of the induction hypothesis, we get

$$\lim_{h\to\infty} Q_h^{\omega_h}(s,a) \geq \lim_{h\to\infty} Q_h^{\pi}(s,a), \tag{35}$$

which is same as

$$Q^{\omega}(s,a) \geq Q^{\pi}(s,a). \tag{36}$$

□

Theorem A.2. *All swarm operators $\mathcal{T}_t^{(i)}$ are contraction mappings[10], i.e.*

$$\|\mathcal{T}_t^{(i)}Q - \mathcal{T}_t^{(i)}Q'\| \leq \alpha\|Q - Q'\|, \tag{37}$$

for some norm.

Proof. We show contraction for supremum norm that is

$$\|\mathcal{T}_t^{(i)}Q - \mathcal{T}_t^{(i)}Q'\|_\infty \leq \alpha\|Q - Q'\|_\infty, \tag{38}$$

for some $\alpha \in [0,1]$ and any $Q, Q' \in \mathcal{Q}$.

$$\begin{aligned}
\|\mathcal{T}_t^{(i)}Q - \mathcal{T}_t^{(i)}Q'\|_\infty &= \max_{s\in S, a\in A} |\mathcal{T}_t^{(i)}Q(s,a) - \mathcal{T}_t^{(i)}Q'(s,a)| \\
&= \max_{s\in S, a\in A} |\gamma E_T(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&= \max_{s\in S, a\in A} \gamma| \sum_{s'\in S} T(s'|s,a)(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&\leq \max_{s\in S, a\in A} \gamma| \sum_{s'\in S} T(s'|s,a) \max_{s'\in S} (Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&= \max_{s\in S, a\in A} \gamma| \max_{s'\in S} (Q(s', \pi^{(i)})_t(s') - Q(s', \pi_t^{(i)}(s'))) \sum_{s'\in S} T(s'|s,a)| \\
&= \max_{s'\in S} \gamma|(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s'))) \times 1| \\
&\leq \max_{s'\in S, a'\in A} \gamma|(Q(s',a') - Q(s',a'))| \\
&= \gamma\|Q - Q'\|_\infty
\end{aligned} \tag{39}$$

Since $\gamma \in [0,1)$, we can see that operator $\mathcal{T}_t^{(i)}$ is a contraction mapping on spuremum norm. □

Theorem A.3. *Swarm RL algorithms are optimality preserving, i.e. for any random initialization $Q_0^i \in \mathcal{Q}$ and subsequent application of swarm operators $\mathcal{T}_t^{(i)}$, the asymptotic Q-functions $\widetilde{Q}^{(i)}, \forall i \in \mathcal{K}$ is same as optimal Q-function Q^*, i.e. for all $s \in S, a \in A$,*

$$Q^*(s,a) = \widetilde{Q}^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)}\mathcal{T}_{t-1}^{(i)}...\mathcal{T}_0^{(i)}Q_0^{(i)}(s,a). \tag{40}$$

Proof. Let $\overline{\widetilde{Q}^{(i)}}$ and $\underline{\widetilde{Q}^{(i)}}$ be the supremum and infimum limits defined as

$$\overline{\widetilde{Q}^{(i)}} = \limsup_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a), \tag{41}$$

and

$$\underline{\widetilde{Q}^{(i)}} = \liminf_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a), \tag{42}$$

respectively.

On subtracting (3) from (15), we get

$$\mathcal{T}_t^{(i)}Q_t^{(i)}(s,a) - \mathcal{T}Q_t^{(i)}(s,a) = \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s')) - \max_b Q_t^{(i)}(s',b)]. \tag{43}$$

Applying limiting infimum conditions to L.H.S,

$$\begin{aligned} L.H.S &= \liminf_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) - \liminf_{t\to\infty} \mathcal{T} Q_t^{(i)}(s,a) \\ &= \underline{\widetilde{Q}^{(i)}}(s,a) - \mathcal{T} \liminf_{t\to\infty} Q_t^{(i)}(s,a) \\ &= \underline{\widetilde{Q}^{(i)}}(s,a) - \mathcal{T}\,\underline{\widetilde{Q}^{(i)}} \end{aligned} \tag{44}$$

Applying limiting infimum conditions to R.H.S

$$\begin{aligned} R.H.S &= \liminf_{t\to\infty} \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s')) - \max_b Q_t^{(i)}(s', b)] \\ &\geq \gamma E_T[\liminf_{t\to\infty} (Q_t^{(i)}(s', \pi_t^{(i)}(s'))) - \liminf_{t\to\infty} (\max_b Q_t^{(i)}(s', b))] \quad \text{(from Fatou's lemma)} \\ &= \gamma E_T[\underline{\widetilde{Q}^{(i)}}(s', \liminf_{t\to\infty} \pi_t^{(i)}(s')) - \max_b \underline{\widetilde{Q}^{(i)}}(s', b)] \\ &= \gamma E_T[\underline{\widetilde{Q}^{(i)}}(s', \operatorname*{argmax}_b \underline{\widetilde{Q}^{(i)}}(s', b)) - \max_b \underline{\widetilde{Q}}^{(i)}(s', b)] \quad \text{(from } t \to \infty \text{ on (16))} \\ &= 0. \end{aligned} \tag{45}$$

Also, from (41) we have

$$\begin{aligned} \overline{\widetilde{Q}^{(i)}}(s,a) &= \limsup_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) \\ &\leq \limsup_{t\to\infty} \mathcal{T} Q_t^{(i)}(s,a) \quad \text{(from comparing (3) and (15))} \\ &= R(s,a) + \gamma[\limsup_{t\to\infty} E_T[\max_b Q_t^{(i)}(s', b)]] \\ &\leq R(s,a) + \gamma E_T[\limsup_{t\to\infty} \max_b Q_t^{(i)}(s', b)] \quad \text{(from Jensen's inequality)} \\ &= R(s,a) + \gamma E_T[\max_b \limsup_{t\to\infty} Q_t^{(i)}(s', b)] \\ &= R(s,a) + \gamma E_T[\max_b \overline{\widetilde{Q}^{(i)}}(s', b)] \\ &= \mathcal{T}\,\overline{\widetilde{Q}^{(i)}}(s,a). \end{aligned} \tag{46}$$

So we get

$$\begin{aligned} \overline{\widetilde{Q}^{(i)}}(s,a) &\leq \mathcal{T}\,\overline{\widetilde{Q}^{(i)}}. \\ \underline{\widetilde{Q}^{(i)}}(s,a) &\geq \mathcal{T}\,\underline{\widetilde{Q}^{(i)}}, \end{aligned} \tag{47}$$

Theorem A.2 shows that all the operators $\mathcal{T}_t^{(i)}$ are a contraction, consequently implying existence of $\lim_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a)$ in (40) and so $\widetilde{Q}^{(i)} = \underline{\widetilde{Q}^{(i)}} = \overline{\widetilde{Q}^{(i)}}$. Therefore from (47), we get

$$\widetilde{Q}^{(i)}(s,a) = \mathcal{T}\,\widetilde{Q}^{(i)}. \tag{48}$$

Since the fixed point value for Bellman operator is unique and equal to Q^*, the above results to $\widetilde{Q}^{(i)} = Q^*$. □

Theorem A.4. *For a swarm RL algorithm, expectation of ensemble regret $Regret_{\pi_t^{(\cdot)}}(s)$ at step t for observed state s is upper-bounded as*

$$E[Regret_{\pi_t^{(\cdot)}}(s)] \leq \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)} \sum_{a\in A} \frac{|\mathcal{J}|\Delta_{s,a}}{e^{\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}}. \tag{49}$$

Proof. Similar to PSRL[3], an important observation to note for Swarm RL is that Q^* and $Q_t^{(i)}$ are identically distributed. Contingent on this, the posterior sampling lemma A.5 adapted from PSRL[3] states the following.

Lemma A.5 (posterior sampling)**.** *If f is the distribution over Q^*, then for any $\sigma(H_t)$ measurable function h ($\sigma(H_t)$ being the σ-algebra generated by the history H_t),*

$$E[h(Q^*)|H_t] = E[h(Q_t^{(i)})|H_t], \quad \forall i \in \mathcal{K}. \tag{50}$$

On taking expectation over H_t, (50) gives $E[h(Q^*)] = E[h(Q_t^{(i)})]$ through the tower property. Using this property on state s and action a on L.H.S of the swarm policy condition (16) gives

$$E[Q_t^{(i)}(s,a)] = E[Q^*(s,a)], \tag{51}$$

and on R.H.S gives

$$\begin{aligned} E[Q_t^{(i)}(s, g_{t-1}^{(i)}(s))] &= E[Q^*(s, g_{t-1}^{(i)}(s))] && \text{(From (51))} \\ &= E[\max_{b\in A} Q^*(s,b)]. \\ &\text{(Since } E[\operatorname*{argmax}_{a\in A} Q^*(s,a)] = E[Q_{t-1}^{(i)}(s, g_{t-1}^{(i)}(s,a)] \text{ using (50))} \end{aligned} \tag{52}$$

From the definition of ensemble regret,

$$\begin{aligned} Regret_{\pi^{(\cdot)}}(s) &= \sum_{i\in\mathcal{K}} \sum_{a\in A} \mathbb{1}\{\pi^{(i)}(s) = a\}\Delta_{s,a} \\ &= \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)} \sum_{a\in A} \mathbb{1}\{\pi_t^{(\mathcal{J})}(s) = a\}|\mathcal{J}|\Delta_{s,a} && \text{(As per (22))} \\ &\text{(Where } \pi_t^{(\mathcal{J})}(s) = \pi_t^{(i)}(s), \quad \forall i \in \mathcal{J}\ ) \\ &\leq \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)} \sum_{a\in A} \mathbb{1}\Big\{ \sum_{j\in\mathcal{J}} (Q_t^{(j)}(s,a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s))) \geq 0\Big\}|\mathcal{J}|\Delta_{s,a} \\ &\text{(From the condition (16)) on swarm update strategy)} \\ &= \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)} \sum_{a\in A} \mathbb{1}\Big\{ \sum_{j\in\mathcal{J}} \chi_t^{(j)}(s,a) \geq 0\Big\}|\mathcal{J}|\Delta_{s,a}, \end{aligned} \tag{53}$$

where $\chi_t^{(j)}$ is a random variable defined as

$$\chi_t^{(j)}(s,a) = Q_t^{(j)}(s,a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s)). \tag{54}$$

From (52) and (51), expectation of $\chi_t^{(j)}(s,a)$ is given by

$$\begin{aligned} E[\chi_t^{(j)}(s,a)] &= E[Q_t^{(j)}(s,a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s))] \\ &= E[Q^*(s,a)] - E[\max_{b\in A} Q^*(s,b)] \\ &= E[Q^*(s,a) - V^*(s)] \\ &= -\Delta_{s,a}. \end{aligned} \tag{55}$$

Lemma A.6 (Chernoff-Hoffending Bound)**.** *Let $Z_1, Z_2, ..., Z_n$ be independent random variables on $\mathcal{R}$ such that $a_i \leq Z_i \leq b_i$ with probability one. If $S_n = \sum_{i=1}^n Z_i$, then for all $t > 0$*

$$Pr(S_n - E[S_n] \geq t) \leq e^{-2t^2/\sum(b_i-a_i)^2} \tag{56}$$

and

$$Pr(S_n - E[S_n] \leq -t) \leq e^{-2t^2/\sum(b_i-a_i)^2}. \tag{57}$$

Using the Chernoff-Hoeffding Inequality, we can bound the expectation of the indicator,

$$\mathbb{1}\{\sum_{j\in\mathcal{J}} \chi_t^{(j)}(s,a) \geq 0\}, \tag{58}$$

as following

$$
\begin{aligned}
E\Big[\mathbb{1}\Big\{\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\geq 0\Big\}\Big] &= Pr\Big(\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\geq 0\Big)\\
&= Pr\Big(\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a) - E\Big[\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\Big] \geq -E\Big[\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\Big]\Big)\\
&= Pr\Big(\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a) - E\Big[\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\Big] \geq |\mathcal{J}|\Delta_{s,a}\Big)\\
&\text{(Using (55))}\\
&\leq e^{\frac{-2(|\mathcal{J}|\Delta_{s,a})^2}{|\mathcal{J}|\times 2^2}}\\
&\text{(Using (56). Range of any } \Delta_{s,a} \text{ is } [-1,1])\\
&= e^{-\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}
\end{aligned}
\tag{59}
$$

Using (59) we can now bound $E[Regret_{\pi^{(\cdot)}(s)}]$ as

$$
\begin{aligned}
E[Regret_{\pi^{(\cdot)}}(s)] &\leq E\Big[\sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)}\sum_{a\in A}\mathbb{1}\Big\{\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\geq 0\Big\}|\mathcal{J}|\Delta_{s,a}\Big]\\
&\text{(Applying expectation on both sides of (53))}\\
&= \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)}\sum_{a\in A}E\Big[\mathbb{1}\Big\{\sum_{j\in\mathcal{J}}\chi_t^{(j)}(s,a)\geq 0\Big\}\Big]|\mathcal{J}|\Delta_{s,a}\\
&\text{(Taking } E \text{ inside summation)}\\
&\leq \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)}\sum_{a\in A}e^{-\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}|\mathcal{J}|\Delta_{s,a}\\
&\text{(From (59))}\\
&= \sum_{\mathcal{J}\in\mathcal{J}_t^{(\cdot)}(s)}\sum_{a\in A}\frac{|\mathcal{J}|\Delta_{s,a}}{e^{\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}}
\end{aligned}
\tag{60}
$$

□

A.1 Open questions

Deciding best target update frequency Too small a target update frequency causes optimistic overestimation (explained in [20]), while too big an update frequency leads to obsolete or vague selection of preferred action sets. Naturally, a way of determining the best update frequency needs to be developed for fastest learning.

Deciding ensemble head count The regret bound (23) can be tightened exponentially by aggressively increasing the number of heads to be much larger than $|A|$, but this increases clustering time complexity exponentialIly as well. If greedy approximate clustering is being used, the accuracy of clusters reduces with increase in number of heads. Therefore, a way of determining the best head count also needs to be developed.

Determining the net ensemble regret bound The regret bound on expected step ensemble regret presented in (23) is parameterized on the policy. Abstracting away the set of preferrable actions that dictates the restrictions on selected policy, we developed the regret bound that allowed us to find the optimal policy for action decisions as well as error computation. A more rigorous unparameterized regret bound on net ensemble regret still needs to be developed for this class of algorithms.

B Containment results for Swarm RL

B.1 Containment of Q-Ensemble Voting

In Q-Ensemble Voting, the action decisions are taken based on majority voting, i.e.

$$a_t = MajorityVote(\{\underset{b\in A}{\text{argmax}}\, Q_t^{(i)}(s_t, b)|\forall i \in \mathcal{K}\}), \tag{61}$$

and based on the observed transitions, the Q-function estimate is updated for every head using the greedy next-state estimates, i.e.

$$Q^{(\cdot)}(s_t, a_t) \leftarrow Q^{(\cdot)}(s_t, a_t) + \alpha(R(s_t, a_t) + \gamma \max_b Q^{(\cdot)}(s_{t+1}, b) - Q^{(\cdot)}(s_t, a_t)). \tag{62}$$

By definition of Swarm RL, an algorithms belongs in this class if it's update equation satisfies the application of Swarm Operator expressed in (15). Selecting $\alpha = 1$, one can see that this update equation becomes

$$Q^{(\cdot)}(s_t, a_t) \leftarrow R(s_t, a_t) + \gamma Q^{(\cdot)}(s_{t+1}, \underset{b\in A}{\text{argmax}}\, Q^{(\cdot)}(s_{t+1}, b)). \tag{63}$$

The next-state policy obeys the requirement of a swarm policy stated in (16), as it is the greedy arg-max action which always belongs to the set of preferable actions. Therefore, Ensemble Voting Q-learning belongs to Swarm RL class of algorithms for $\alpha = 1$.

B.2 Containment of Bootstrapped Q-learning

DQNs are adaptations of Q-learning, with heuristics for faster training speed and therefore instead of trying to find whether Bootstrapped DQN belongs to Swarm RL, we would show that the simplified Q-learning version, aka. Bootstrapped Q-learning, belongs to Swarm RL.

For Bootstrapped Q-learning, the action decision is given by a randomly sampled head's greedy arg-max choice, i.e.

$$a_t = \underset{b}{\text{argmax}}\, Q^{(i)}(s_t, b), \tag{64}$$

where head i is sampled from a uniform distribution, $Uniform(\mathcal{K})$, at the start of every episode.

The Q-function update depends on a probability p Bernoulli mask vector $M = (m^{(1)}, m^{(2)}, ..., m^{(|\mathcal{K}|)})$, according to which the heads are updated as

$$Q^{(j)}(s_t, a_t) \leftarrow Q^{(j)}(s_t, a_t) + \alpha(R(s_t, a_t) + \gamma \max_b Q^{(j)}(s_{t+1}, b) - Q^{(j)}(s_t, a_t)), \tag{65}$$

for all j that is unmasked in vector M, i.e. $m^{(j)} = 1$.

Osband et al. [8] concluded that best results were seen when Bernoulli probability p was one, i.e when nothing was masked. This makes the above update equation identical to (62), and therefore also belongs to Swarm RL class using the same argument followed for Q-Ensemble Voting.

C Technical appendix

C.1 MDP formulation, parameters and neural net architecture

The comparison was conducted on a linear MDP and it's 2D extension as shown in figure 3 and 6 respectively. The first MDP toy problem, introduced in Osband et al. [8] to depict deep exploration of Bootstrapped DQN, consists of $n = 15$ states connected in a linear fashion, with first and last states being the only rewarding as well as terminal states, rewards being $r = 1/1000$ and $r = 1$ respectively. Every episode begins at state $s = 2$ with two actions available at every non terminal state and ends when a terminal state has been reached or after 18 transitions, whichever comes first.

The second MDP poses an even difficult problem, involving 9×9 states and four actions per non terminal state. This MDP also has only two rewarding and terminal states, $s_{1,1}$ with reward $r = 1/9$ and $s_{9,9}$ with reward $r = 1$. Each episode starts at state $s_{2,2}$ and terminates when a terminal state is reached or after 81 transitions, whichever comes first.

One hot state encoding was followed for both the MDPs, i.e. feature mapping $\phi(s) := (\mathbb{1}\{x = s\})$ in $\{0, 1\}^n$, n being 15 and 81 respectively. Figure 4 shows a comparison of 100 episode running average of net returns between Ensemble Voting DQN, Bootstrapped DQN and Optimal Swarm DQN averaged over 5 different seeds. Table 1 states the parameters used for training for the Optimal Swarm DQN. Following exploration epsilon schedule were used: $1 \rightarrow 0.01$ in $80,000$ steps for the first MDP and in $200,000$ steps for the second MDP.

Parameter Name	Value
Number of Steps	400,000
Experience Replay Size	100,000
Training Begin Step	5,000
Target Update Frequency	750
Sample Batch Size	64
Memory Sample Frequency	1
Optimizer	Adam
Adam beta1	0.9
Adam beta2	0.99
Adam epsilon	0.0001
Adam Learning Rate	0.0001
Ensemble heads	5

Table 1: Parameter table for Optimal Swarm DQN

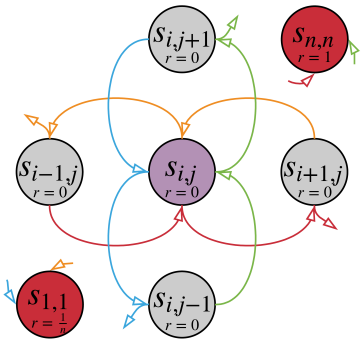

Figure 6: 2D MDP toy problem

Identical neural network architecture was used for Ensemble Voting DQN, Bootstrapped DQN and Optimal Swarm DQN, and consisted of one fully connected shared layer with 64 hidden nodes and one fully connected private layer per head, again with 64 hidden nodes, as shown in figure 7.

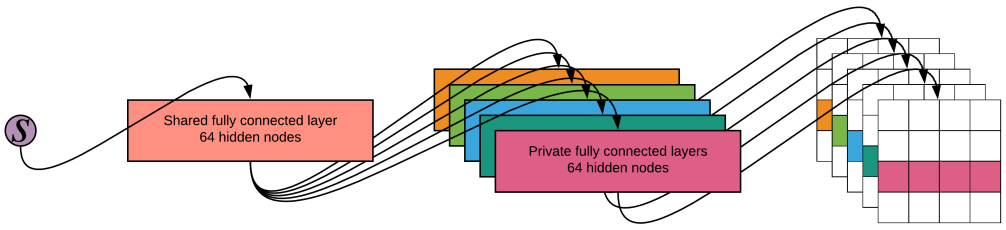

Figure 7: Neural network architecture with $|\mathcal{K}| = 5$ heads

C.2 Greedy approximation for optimal clustering

Here we describe the modified version of Greedy Set Cover algorithm, optimized for performing the *Agglomerate* function used in algorithm 1 for a batch of observations. Instructions with unbounded variables i, j can be parallelized and improves training time considerably.

Algorithm 2 Parallelized Greedy Clustering

procedure GREEDY-AGGLOMERATE(Observation batch $s[\cdot]$, Target $T^{(\cdot)}$, Current $Q^{(\cdot)}$)

$QCurrent[i] \leftarrow Q^{(\cdot)}(s[i])$
$QTarget[i] \leftarrow T^{(\cdot)}(s[i])$
$QTargetBestAction[i] \leftarrow argmax(QTarget[i], axis = 1)$
$PrefActionSet[i][j] \leftarrow QCurrent[i][j][\cdot] \geq QCurrent[i][j][QTargetBestAction[i][j]]$

$B \leftarrow size(s)$
$NoAgent \leftarrow zeros(B, |\mathcal{K}|)$
$NotCovered \leftarrow ones(B, |\mathcal{K}|)$
$Cover \leftarrow zeros(B, 1, |A|)$

while $NotCovered != NoAgent$ **do**

$AllowedActions \leftarrow PrefActionSet \odot repeat(NotCovered, [1, 1, |A|])$
$GreedyActions \leftarrow argmax(sum(AllowedActions, axis = 1), axis = 1)$
$GreedyActionCovered[i] \leftarrow PrefActionSet[i][\cdot][GreedyActions[i]]$
$AlreadyCovered[i] \leftarrow all(NotCovered \odot GreedyActionCovered[i] < 1)$
$Cover \leftarrow Cover + (1 - AlreadyCovered) \odot onehot(GreedyActions, size = |A|)$
$NotCovered \leftarrow NotCovered \odot (1 - GreedyActionCovered)$

return $Cover, PrefActionSet$

This implementation can be further improved in several ways, such as by tampering with internal data structures or looping over each batch elements separately in parallel rather than what is done above. Furthermore, state of the art optimal Set Cover approximation algorithms can be explored for the purpose. We ignored such improvements due to increase in complexity and difficulty in implementation for a small increment in presumed running time.