

Federated Reinforcement Learning at the Edge: Exploring the Learning-Communication Tradeoff

Konstantinos Gatsis

Abstract—Modern cyber-physical architectures use data collected from systems at different physical locations to learn appropriate behaviors and adapt to uncertain environments. However, an important challenge arises as communication exchanges at the edge of networked systems are costly due to limited resources. This paper considers a setup where multiple agents need to communicate efficiently in order to jointly solve a reinforcement learning problem over time-series data collected in a distributed manner. This is posed as learning an approximate value function over a communication network. An algorithm for achieving communication efficiency is proposed, supported with theoretical guarantees, practical implementations, and numerical evaluations. The approach is based on the idea of communicating only when sufficiently informative data is collected.

I. INTRODUCTION

Recent years have seen a shift in cyber-physical system architectures from systems running in isolation or local networks to systems connected to the cloud for outsourcing data and computations. This has given rise to the realization that the resulting bandwidth and communication requirements can become a bottleneck, and as a result, computing at the edge of the architecture is explored. Toward this end, this paper introduces a new framework for performing distributed reinforcement learning from data collected locally at individual agents/robots at the edge.

In the area of control of distributed and multi-agent systems, very recent developments include algorithms for multi-agent reinforcement learning [1], [2], reinforcement learning over networks/graphs [3]–[6], as well as the search for appropriate parameterizations for these problems [7]. However when dealing with distributed learning at the edge, there is also the need for communication efficiency, especially if agents have high dimensional time-series data and operate over resource-limited communication networks. The bulk of distributed and communication-efficient learning, also termed federated learning, is focused on static machine learning problems, such as classification. To overcome the communication bottleneck gradients of the objective with respect to the parameters being learned are sent, instead of the data itself [8]–[15]. For the problem of communication-efficient reinforcement learning, only now initial approaches emerge, including non-periodic updates for policy gradient methods [16], distributed peer-to-peer network architectures [17], specific problems such as multi-armed bandits [18], or regret analysis of online distributed reinforcement learning [19].

The author is with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK. Email: konstantinos.gatsis@eng.ox.ac.uk.

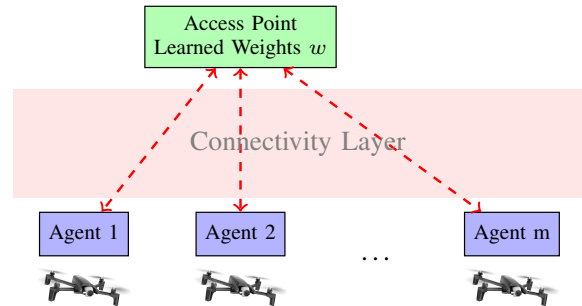


Fig. 1: Architecture for distributed reinforcement learning. Agents collect state transition and cost data and collaboratively learn a value function with the aid of an access point/server. Communication efficiency is achieved by uploading only when data are informative.

This paper introduces a new approach for communication-efficient reinforcement learning problems. To avoid costly exchange of time-series data, agents should assess *how informative* are the data they collect, and communicate if the update will help the process of learning. The setup, described in Section II and shown in Fig. 1, involves multiple agents collecting state transitions and costs in a distributed manner for the purpose of learning approximate value functions, which is a central problem in reinforcement learning. Specifically linear value function approximation is considered. Compared to recent approaches [16]–[19] the main novelty of this approach is that the communication cost is part of the performance criterion, and the resulting algorithms *by design* explore efficiently the tradeoff between communication and learning. This is obtained both theoretically (Section III) as well as in numerical examples in finite and continuous state spaces (Section V).

The approach adapts ideas from the author’s recent work on communication-efficient linear regression [12], [13] to the reinforcement learning problem. Technically the approach is different here as, to ensure convergence close to an optimal solution, agents start by sending updates only when very informative data are found and proceed at later iterations with sending more updates when less informative data are found. More broadly, the technical methodology relates to the problem of efficient control over networks and event-triggered control [20], [21], event-triggered learning [22], [23], resource-aware optimization [24], [25], and adaptive filtering over graphs [26], [27].

II. PROBLEM SETUP

The architecture examined in this paper, shown in Fig. 1, involves an access-point/server solving a reinforcement

learning task on data collected by multiple agents. Specifically, each agent collects independently state transitions (from a state x^t to a new state x_+^t) and cost measurements (of the form c^t at state x^t), and communicates with the server in order to jointly learn an approximate value function $V(x)$ parameterized by a vector of weights w . The aim is to achieve this with communication efficiency, i.e., without agents communicating all collected data all the time. An example scenario is presented, and the formal mathematical formulation follows next.

Example 1 (Motivating scenarios). As a first example, suppose the agents in Fig. 1 are robots/drones/vehicles with identical system dynamics. Then they can each collect data from their own trajectories (state transitions) as well as costs measuring how well they can achieve a common task, such as tracking a desired behavior. Transmitting all these data over the communication network would be costly. A recent case study of this form by Google [28] considers the large scale training of control and manipulation policies using multiple robots collecting image data concurrently. As a second example, suppose the agents in Fig. 1 are sensors/actuators in an industrial setting collecting time-series data from multiple identical physical processes, for the purpose of improving the control of the processes.

A. Value Function Learning

We assume a Markov Decision Process problem defined by the state space X , the action space A , the transition probabilities $\mathbb{P}(x_+ | x, a)$ ¹, and a cost function $c : X \times A \rightarrow \mathbb{R}$ giving rise to costs of the form $c_k = c(x_k, a_k)$. We are interested in approximating the value function, or cost-to-go function of a given policy $a_k = \pi(x_k)$, defined as $V(x) := \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k c(x_k, a_k) | x_0 = x]$ with a discount factor $\gamma \in (0, 1)$. A general theoretical approach for finding the value function is to perform the Value Iteration Algorithm, which iteratively fixes a current guess of the value function $V^{\text{current}}(x)$, $x \in X$, and updates it according to

$$V^{\text{updated}}(x) \leftarrow c(x, \pi(x)) + \gamma \mathbb{E}[V^{\text{current}}(x_+) | x, a = \pi(x)] \quad (1)$$

for all points $x \in X$, then resets

$$V^{\text{current}}(x) \leftarrow V^{\text{updated}}(x) \quad (2)$$

for all points $x \in X$, and repeats the process. Under technical conditions [29, Vol. II, Ch. 2], this process can converge to the desired value function $V(x)$, which is a fixed point of (1) (Bellman equation).

In reinforcement learning and approximate dynamic programming, this iteration is performed approximately, using data collected by following the given policy. The data consists of multiple state transitions from states $x^t, t = 0, 1, 2, \dots$ to new states $x_+^t, t = 0, 1, 2, \dots$, following the above transition probabilities, and stage costs of the form c^t at state $x^t, t = 0, 1, 2, \dots$. In general we can denote these as tuples $(x^t, c^t, x_+^t), t = 0, 1, 2, \dots$. In practice, these state

transitions can be just segments from longer state trajectories. The state samples $x^t, t = 0, 1, 2, \dots$ are modeled as drawn from a distribution $d(x)$ in this paper.

Moreover, instead of computing the updated value function in (1) in the space of all functions $\{V : X \rightarrow \mathbb{R}\}$, a restricted function class is selected. In this paper we follow the commonly employed linear function class [30]. In other words, we are trying to explain $V^{\text{updated}}(x)$ in (1) as a linear combination of basis functions (features) as

$$V^{\text{updated}}(x) \approx \sum_{i=1}^n w_i \phi_i(x). \quad (3)$$

Here $w \in \mathbb{R}^n$ if the vector of weights (linear combination) to be learned, consisting of elements w_i , and $\phi_i(x)$ are fixed basis functions (not learned). Examples of basis functions, especially when X is a real vector space, include discretization functions, polynomial functions, radial basis functions, and others [29], [30].

To pick one particular approximation in (3), we are interested in minimizing the squared error

$$\underset{w \in \mathbb{R}^n}{\text{minimize}} J(w) = \mathbb{E}_d \left[V^{\text{updated}}(x) - \sum_{i=1}^n w_i \phi_i(x) \right]^2 \quad (4)$$

where the expectation is with respect to a distribution $d(x)$, which corresponds to the distribution of the data collected by the agents. To sum up, instead of updating the value function according to (1), we instead approximately update to the function $\sum_{i=1}^n w_i^* \phi_i(x)$ where w^* is the vector solving (4). After this approximation is found, the process (1) repeats. The obtained approximation takes the place of the current value function V^{current} , and a updated value function V^{updated} is found by a new approximation. Equivalently this can be thought as fixing a current vector w , finding a new vector w , and repeating.

A fundamental question then is how to solve problem (4) from data. An approach is to perform what can be considered a stochastic gradient descent. Specifically, this in an iterative algorithm of the form

$$w_{k+1} = w_k - \varepsilon \hat{\nabla} J(w_k), \quad k = 0, 1, \dots \quad (5)$$

where $\varepsilon > 0$ is a stepsize, and the gradient is approximated from the data tuples as

$$\hat{\nabla} J(w_k) = \frac{1}{T} \sum_{t=0}^T \phi(x^t) (w_k^T \phi(x^t) - c^t - \gamma V^{\text{current}}(x_+^t)). \quad (6)$$

In this expression, k refers to iterations of the algorithm (5), while t refers to identically distributed samples of state transitions and costs from the Markov Decision Process, and there are T of them in total. It can be verified that this yields an unbiased estimate of the gradient of (4). After a large number of iterations N of (5), which also means after drawing many data samples, and with an appropriate stepsize explained below, the iterate w_N will converge close to the optimal solution of the problem (4) which we denote as w^* .

Remark 1 (Relation to other Reinforcement Learning approaches). In this paper we illustrate how the above value

¹or alternatively general state dynamics of the form $x_+ = f(x, a, w)$ for some noise variable w

function approximation in (4) can be performed with communication efficiency. This approximation then needs to take place at each iteration of the value iteration algorithm (1) – see also Algorithm 1. The approach can also be extended to learn a Q-function approximation but this is not further discussed in this paper due to limited space. More broadly, these algorithms are variants of Projected Value Iteration [29, Vol. II, Ch. 6], and are behind many successful reinforcement learning approaches [31], [32]. These algorithms are attractive because under technical conditions they converge to a unique point.

B. Communication-efficient reinforcement learning problem

Given the above modeling for a reinforcement learning task that needs to be solved, the communication problem is as follows. At each iteration k , the server broadcasts the current weights w_k to all agents. Then each agent i collects T local data samples identically distributed (across samples and across agents), computes a local stochastic gradient $\hat{\nabla}_i J(w_k)$ from the available local data using formula (6), and decides whether to transmit this gradient update over the communication network to the receiving server. The server updates the current vector of weights w_k depending on the information received from different agents. For simplicity of exposition the case of two agents is considered and theoretically analyzed, and experiments with more agents are conducted numerically in Section V. This leads to the update rule at the server

$$w_{k+1} = \begin{cases} w_k - \varepsilon \hat{\nabla}_1 J(w_k) & \text{if 1 transmits} \\ w_k - \varepsilon \hat{\nabla}_2 J(w_k) & \text{if 2 transmits} \\ w_k - \frac{\varepsilon}{2} (\hat{\nabla}_1 J(w_k) + \hat{\nabla}_2 J(w_k)) & \text{if both transmit} \\ w_k & \text{if no transmits} \end{cases} \quad (7)$$

We further denote with $\alpha_k^i \in \{1, 0\}$ the decision for each agent $i = 1, 2$ to transmit or not.

At the next iteration $k+1$ a new set of data is collected at each agent, a new stochastic gradient direction $\hat{\nabla}_i J(w_{k+1})$ with respect to the new vector w_{k+1} is computed at each agent, and the process repeats until a final iteration N . The algorithm is also described in Algorithm 1.

The aim will be to *avoid sending updates all the time in order to limit the communication burden*. Hence we consider the average communication cost during all iterations and across all agents as

$$\frac{1}{N} \sum_{k=0}^{N-1} \frac{\alpha_k^1 + \alpha_k^2}{2} \quad (8)$$

On the other hand, we want to understand the progress of learning, hence we also measure how well the final set of weights w_N solves (4), in other words we measure the cost function $J(w_N)$.

We note that since the data points at each iteration and at each agent are random, so are the constructed stochastic gradient directions, and so are the vectors w_k , and so are the decisions α_k^i of the agents to transmit. We propose then to measure the efficiency of the implementation *on average*

Algorithm 1 Distributed Approximate Value Iteration Algorithm

- 1: Fix a policy π at all agents
 - 2: Initialize value function guess $V^{\text{current}}(x)$, e.g., with w^{current} , at all agents.
 - 3: Initialize weights w_0 at server
 - 4: **for** Iteration $k = 0, 1, 2, \dots, N-1$ **do**
 - 5: Server transmits weights w_k to agents
 - 6: At each agent i collect T state transition and cost samples following policy π
 - 7: At each agent i compute stochastic gradient $\hat{\nabla}_i J(w)$
 - 8: At each agent i implement communication (10) with the approximation (16)
 - 9: At the server update weights w_{k+1} according to (7)
 - 10: **end for**
 - 11: Replace $V^{\text{current}}(x) \leftarrow \sum_{i=1}^n w_N(i) \phi_i(x)$ using the final weights w_N and go to Step 3
-

over the data points collected. As a result, we establish the performance metric

$$\mathbb{E}_{\text{data}} \left[\lambda \sum_{k=0}^{N-1} \frac{\alpha_k^1 + \alpha_k^2}{2N} + J(w_N) \right] \quad (9)$$

Here $\lambda > 0$ is a tuning parameter that is used to either penalize communication or learning performance. It is worth emphasizing then that there are two expectations in this paper. One is the integral defined in the objective $J(w)$ in (4), and another is the expected performance of the implementation in (9) computed as an integral over tuples x^t, c^t, x_+^{t+1} – in total $2NT$ tuples over agents, iterations, and samples – and denoted as \mathbb{E}_{data} to clarify the difference.

III. THEORETICAL SCHEME FOR COMMUNICATION-EFFICIENT REINFORCEMENT LEARNING

The approach is based on the notion of performance gain which can be thought as a measure of how informative are the data collected at each agent at each time step with respect to the reinforcement learning problem. The gain at agent $i = 1, 2$ can be calculated by measuring how much will the objective change if the agent sends the update. Whether this gain is negative or positive depends on the random direction of the update. The proposed approach then is to send a gradient update if the gain is large enough. Mathematically we write

$$\alpha_k^i = \begin{cases} 1 & \text{if } J(w_k - \varepsilon \hat{\nabla}_i J(w_k)) - J(w_k) \leq -\frac{\lambda}{\rho^{N-1-k}} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where the scalar parameter $\lambda > 0$ is the one defined in the performance criterion, and $\rho \in (0, 1)$ is a parameter – whose impact is further discussed in the theoretical results below. Intuitively this approach saves up communication resources, because the updates will be infrequent – see also related federated learning approaches in [13]–[16]. Moreover, the term at the right hand side, which measures how informative are the data at the current iteration, is decreasing (in absolute value) as the number of iterations grow. Hence, at the

beginning only very informative updates are transmitted, while as learning progresses, less informative updates are transmitted as well.

Assumption 1. The $n \times n$ symmetric matrix $\mathbb{E}_d \phi(x) \phi(x)^T$ is positive definite.

This assumption guarantees the solution to the main problem (4) is unique.

Assumption 2. The step size $\varepsilon > 0$ in (7) satisfies

$$|1 - 2\varepsilon \mu_i(\mathbb{E}_d \phi(x) \phi(x)^T)| < 1 \quad (11)$$

for all eigenvalues of the matrix at the right hand side.

This assumption guarantees the step size is small enough so that convergence, even without communication constraints, is satisfied. A sufficient condition is $\varepsilon < 2/\mu_{\max}$ where μ_{\max} is the largest eigenvalue of the above matrix. Assumption 1 guarantees that (11) can be met.

Assumption 3. The parameter ρ in (10) satisfies

$$\rho \geq \max_i (1 - 2\varepsilon \mu_i(\mathbb{E}_d \phi(x) \phi(x)^T))^2 \quad (12)$$

for all eigenvalues of the matrix at the right hand side.

This assumption states that measuring the informativeness of the data should decrease at a sufficiently slow rate. Assumptions 2 and 3 together guarantee that $\rho < 1$ is a possible choice.

The main theoretical result of this paper is established next. The proof can be found in an extended version [33].

Theorem 1 (Communication-Efficient Value Function Approximation). *Consider the optimization problem defined in (4). Consider the update rule in (7). Suppose $\hat{\nabla}_i J(w_k), i = 1, 2$, are independent random variables with mean equal to $\nabla J(w_k)$ and covariance G at each iteration k . Consider the communication strategy in (10) with a fixed number of iterations N . Let Assumptions 1-3 hold. Then we have that*

$$\mathbb{E}_{data} \left[\lambda \sum_{k=0}^{N-1} \frac{\alpha_k^1 + \alpha_k^2}{2N} + J(w_N) \right] \leq \lambda + J(w^*) + \rho^N [J(w_0) - J(w^*)] + \frac{1 - \rho^N}{1 - \rho} \varepsilon^2 \text{Tr}(\mathbb{E}_d \phi(x) \phi(x)^T G) \quad (13)$$

where w^* is the optimal solution of (4) and the expectation is with respect to the data collected until iteration N .

We have the following observations. The terms on the right hand side of (13) measure the suboptimality of learning, including the terms that have to do with the poor initialization $J(w_0) - J(w^*)$ and the noise of the gradients captured by the matrix G . As the number of iterations N grows, the impact of the former is diminished, and only the latter remains. This expression also suggests that ideally one would pick the parameter ρ to be the minimum allowed by the above Assumption 3.

The theorem characterizes the tradeoff between communication and learning. By increasing the parameter λ , there is a higher penalty for communicating, and as a result, the

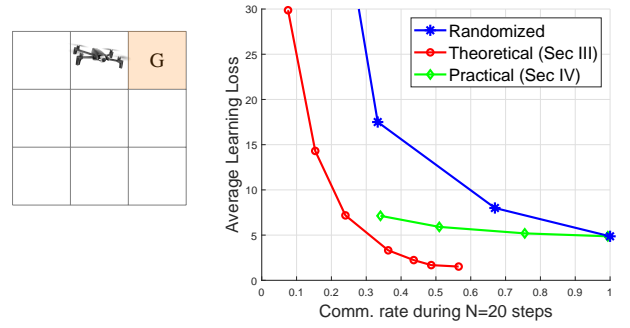


Fig. 2: Left: Grid example. Agents count the time it takes to reach the goal G while there is 50% uncertainty in transitioning to the right at top row. Right: Comparison between our communication approach in (10) requiring the model to compute the performance gains versus estimating the gains by (16).

agents communicate less often – it is possible to state this fact formally [13]. As a consequence, learning performance will be impacted. But the above theorem *guarantees by design* that there will be a graceful compromise between communication performance and learning – the left hand side (13) cannot be arbitrarily poor.

It is worth noting that, inspecting the right hand side of (13), the theorem states that the aggregate performance will be better than having only one agent transmitting all the time. It is desirable to extend the analysis to get bounds sharper than (13), which will be the topic of future work.

Remark 2. In Theorem 1 we assumed for simplicity that the stochastic gradients have bounded covariances that are constant over time. In reality for the problem above the covariance of the stochastic gradient in (6) will depend on the current iterate w_k , but our choice can be justified. For example we can add a projection to a bounded set $\|w_k\| < M$ in the algorithm, so we only search over this restricted set of weights, resulting in bounded gradient noise covariances.

Remark 3 (Analysis over multiple iterations). The above theoretical analysis is performed for one iteration of Alg. 1 (lines 3-10). As a result, at each iteration communication-efficiency is guaranteed. After many iterations, the algorithm will converge to a neighborhood of the desired set of weights w , i.e., the value function approximation. This analysis will be explored in future work.

IV. PRACTICAL SCHEME FOR COMMUNICATION-EFFICIENT REINFORCEMENT LEARNING

Implementing the theoretical communication scheme in (10) is practically impossible because it requires information that is not known. Specifically it would require for every agent to know the Markov Decision Process and the data distribution in order to compute the actual performance gain. Since these are unknown, one approach is to *estimate the performance gain from the data*. In particular, since the objective function is quadratic, we can write the performance

gain as

$$J(w_k - \varepsilon \hat{\nabla}_i J(w_k)) - J(w_k) = -\varepsilon \hat{\nabla}_i J(w_k)^T \nabla J(w_k) + \frac{1}{2} \varepsilon^2 \hat{\nabla}_i J(w_k)^T \nabla^2 J(w_k) \hat{\nabla}_i J(w_k) \quad (14)$$

This is a quadratic function of the stochastic gradient $\hat{\nabla}_i J(w_k)$. Then we can approximate the gradient and the Hessian at each agent as

$$\begin{aligned} \nabla J(w_k) &\approx \hat{\nabla}_i J(w_k) \\ \nabla^2 J(w_k) &\approx \frac{1}{T} \sum_{t=0}^T \phi(x^t) \phi(x^t)^T \end{aligned} \quad (15)$$

Hence, using the expression for the information gain in (14), we can approximate the gain at agent $i = 1, 2$ as²

$$\begin{aligned} J(w_k - \varepsilon \hat{\nabla}_i J(w_k)) - J(w_k) &\approx \\ -\hat{\nabla}_i J(w_k)^T \left[I - \varepsilon \frac{1}{2} \frac{1}{T} \sum_{t=0}^T \phi(x^t) \phi(x^t)^T \right] \hat{\nabla}_i J(w_k)^T \end{aligned} \quad (16)$$

It is important to emphasize that this is no longer a simple quadratic function of the stochastic gradient but a more complicated function - we note that the data appear both in the stochastic gradients as well as in the matrix in the middle. This approximate value of the gain may take again positive or negative values but it induces an approximation error/bias.

As a result, we can implement the communication decision in (10) with the approximation in (16). In this case we no longer have the performance guarantee in Theorem 1. In numerical evaluations however we see that despite the bias this mechanism performs very well.

Remark 4. A different perspective is followed by [16]. When agents do not update their gradients at the server, the server keeps a memory of past received gradients and uses them for gradient descent in (7). The advantage of the present approach is that it introduces an explicit communication-learning tradeoff that can be controlled by the parameter λ . Technically, the approach [16] is developed for policy gradient methods, while here the approach is developed for value-function-based approaches.

V. NUMERICAL RESULTS

We consider first a grid exploration example (Fig. 2) which is a finite state Markov Decision Process with state space X . An agent can move in four directions subject to the boundary constraints, while at the top part of the grid there is a 50% uncertainty in transitions to the right because of disturbances. The objective function is the total undiscounted ($\gamma = 1$) time it takes to reach a desired goal location G. We collect measurements from multiple agents following the policy that randomizes over all actions at each state.

We perform one iteration of Algorithm 1 following (1) where the initial value function is chosen randomly.

²Overall at each agent these computations require $O(Tn)$ operations hence are scalable.

In order to learn the exact value function we take the basis functions/features to be the indicators $\phi(1) = [1, 0, \dots, 0]^T$, $\phi(2) = [0, 1, 0, \dots, 0]^T$, etc in (3). We suppose each agent has few data tuples $T = 10$ in every iteration, as described in Section II, chosen from a uniform distribution d on this finite state space, and we take the stepsize to be $\varepsilon = 1$. We implement the theoretical approach (10) and the practical approach using (16). We pick the parameter ρ to be close to its smallest value allowed by Assumption 3. For varying values of the parameter λ we empirically compute the average communication rate (8), and the average final learning performance $J(w_N)$ as defined in (4). The achieved tradeoff between the two is shown in Fig. 2. As a comparison, we also include the results when agents randomly decide whether to transmit their gradients or not, leading to an inefficient outcome.

For the theoretical algorithm (10), we observe that a very high communication efficiency can be achieved, i.e., by transmitting only a fraction of the time it is possible to find very good solutions to the value function approximation problem, hence achieving a good communication-learning tradeoff. For the practical approach using (16) of course the learning loss is higher due to the bias introduced, but we also clearly observe a very good tradeoff between all agents communicating all the time versus scarce communications only when necessary.

Then for a more complicated scenario we consider a continuous (uncountable) state space problem $X = \mathbb{R}^2$. State transitions are given by a stochastic linear dynamical system $x_+ = Ax + w$, with $A = \begin{bmatrix} 0.8 & -0.2 \\ 0.1 & 1 \end{bmatrix}$, where w is a zero-mean Gaussian noise variable with covariance 0.1. We also consider a quadratic cost $c(x) = \|x\|^2$ and a discount factor $\gamma = 0.9$. We want to perform one value iteration following (1) where the initial value function is chosen randomly. We would like to approximate the updated value function per (3) as a linear combination of polynomial functions of the states with maximum degree 2 - that is $\phi(x) = [x_1^2, x_2^2, x_1 x_2, x_1, x_2, 1]^T \in \mathbb{R}^6$ - and we are searching for a vector of weights $w \in \mathbb{R}^6$. We suppose each agent has at every iteration $T = 10^3$ data tuples, as described in Section II, chosen from a uniform distribution d on the space $[0, 1]^2$. We take the stepsize to be $\varepsilon = 1$ and the parameter $\rho = 0.999$. As expected when learning value functions on continuous state space, in this case we need many more iterations and a larger amount of data, in the order of 10^8 data tuples.

We implement the practical communication algorithm with (16). Fig. 3a shows the evolution of weights for a large penalty λ on communication rate. The figure shows that at the beginning no communication takes place because not very high informative gradients are found, while more communication takes place as learning progresses. Fig. 3b shows the same setup with a smaller penalty λ on communication, and we observe higher communication rate and faster learning of the weights. These observations verify the findings of Theorem 1. Finally Fig. 3c shows the same setup but with a larger number of agents, 10 instead of 2. We observe that learning happens faster, with almost the same

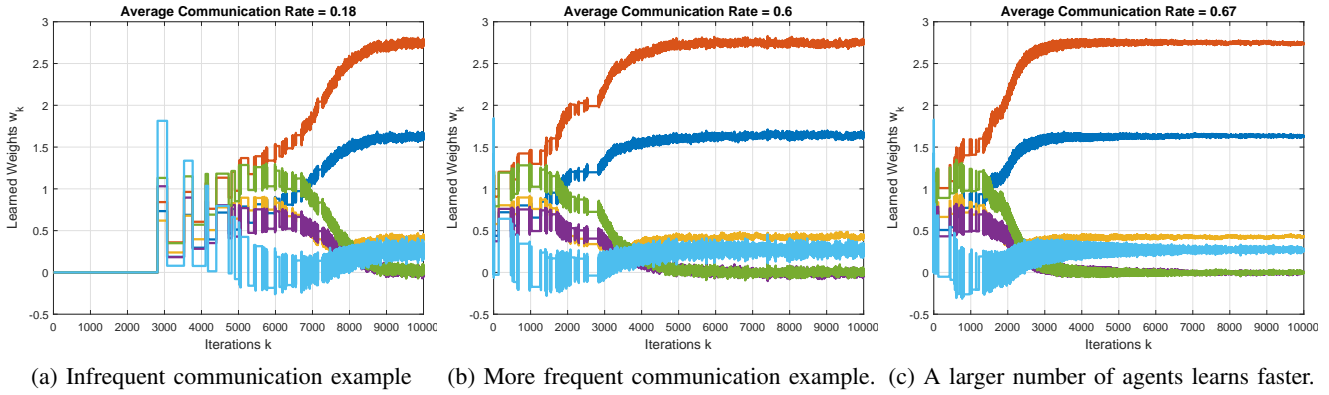


Fig. 3: Communication-Efficient Distributed Value Function Approximation

amount of average communication rate. This observation is not yet studied theoretically here and will be explored in future work.

REFERENCES

- [1] G. Qu, A. Wierman, and N. Li, "Scalable reinforcement learning of localized policies for multi-agent networked systems," in *Learning for Dynamics and Control*, pp. 256–266, PMLR, 2020.
- [2] S. Fattahi, N. Matni, and S. Sojoudi, "Efficient learning of distributed linear-quadratic control policies," *SIAM Journal on Control and Optimization*, vol. 58, no. 5, pp. 2927–2951, 2020.
- [3] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*, pp. 5872–5881, PMLR, 2018.
- [4] Y. Zhang and M. M. Zavlanos, "Distributed off-policy actor-critic reinforcement learning with policy consensus," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4674–4679, IEEE, 2019.
- [5] L. Cassano, K. Yuan, and A. H. Sayed, "Multi-agent fully decentralized value function learning with linear convergence rates," *IEEE Transactions on Automatic Control*, 2020.
- [6] T. T. Doan, S. T. Maguluri, and J. Romberg, "Finite-time performance of distributed temporal-difference learning with linear function approximation," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 1, pp. 298–320, 2021.
- [7] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on Robot Learning*, pp. 671–682, PMLR, 2020.
- [8] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.
- [9] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2018.
- [10] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031, 2020.
- [11] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, 2019.
- [12] K. Gatsis, "Adaptive scheduling for machine learning tasks over networks," in *American Control Conference (ACC)*, 2021.
- [13] K. Gatsis, "Linear regression over networks with communication guarantees," in *3rd Annual Learning for Dynamics & Control Conference*, 2021.
- [14] H. S. Ghadikolaei, S. Stich, and M. Jaggi, "Lena: Communication-efficient distributed learning with self-triggered gradient uploads," in *International Conference on Artificial Intelligence and Statistics*, pp. 3943–3951, PMLR, 2021.
- [15] N. Singh, D. Data, J. George, and S. Diggavi, "Sparq-sgd: Event-triggered and compressed communication in decentralized optimization," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3449–3456, IEEE, 2020.
- [16] T. Chen, K. Zhang, G. B. Giannakis, and T. Basar, "Communication-efficient policy gradient methods for distributed reinforcement learning," *IEEE Transactions on Control of Network Systems*, 2021.
- [17] D. J. Ornig and M. Mazo Jr, "Event-based communication in multi-agent distributed q-learning," *arXiv preprint arXiv:2109.01417*, 2021.
- [18] A. Mitra, H. Hassani, and G. Pappas, "Robust federated best-arm identification in multi-armed bandits," *arXiv preprint arXiv:2109.05700*, 2021.
- [19] M. Agarwal, B. Ganguly, and V. Aggarwal, "Communication efficient parallel reinforcement learning," in *Uncertainty in Artificial Intelligence*, pp. 247–256, PMLR, 2021.
- [20] M. Eisen, M. M. Rashid, K. Gatsis, D. Cavalcanti, N. Himayat, and A. Ribeiro, "Control aware radio resource allocation in low latency wireless control systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7878–7890, 2019.
- [21] M. H. Mamduhi, D. Maity, J. S. Baras, and K. H. Johansson, "A cross-layer optimal co-design of control and networking in time-sensitive cyber-physical systems," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 917–922, 2020.
- [22] F. Solowjow, D. Baumann, J. Garcke, and S. Trimpe, "Event-triggered learning for resource-efficient networked control," in *2018 Annual American Control Conference (ACC)*, pp. 6506–6512, 2018.
- [23] S. Ghosh and V. Gupta, "Eventgrad: Event-triggered communication in parallel stochastic gradient descent," in *2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (A4S)*, pp. 1–8, IEEE, 2020.
- [24] S. Magnússon, H. Shokri-Ghadikolaei, and N. Li, "On maintaining linear convergence of distributed learning and optimization under limited communication," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6101–6116, 2020.
- [25] S. Magnússon, C. Enyioha, N. Li, C. Fischione, and V. Tarokh, "Convergence of limited communication gradient methods," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1356–1371, 2017.
- [26] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. ARTICLE, pp. 311–801, 2014.
- [27] J.-W. Lee, S.-E. Kim, and W.-J. Song, "Data-selective diffusion lms for reducing communication overhead," *Signal Processing*, vol. 113, pp. 211–217, 2015.
- [28] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4–5, pp. 421–436, 2018.
- [29] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 2012.
- [30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [31] M. Riedmiller, "Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, pp. 317–328, Springer, 2005.
- [32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [33] K. Gatsis, "Federated reinforcement learning at the edge," *arXiv preprint arXiv:2112.05908*, 2021.