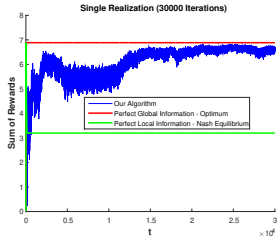
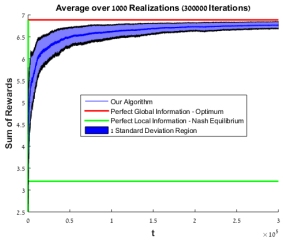
We thank the reviewers for their detailed and constructive comments, especially during these challenging times.

Numerical Validation (R1,R3,R4): We will add the figures below, displaying the typical behavior of a realization and the statistical behavior over time. This also shows that the convergence time is reasonable for this challenging problem with limited information. For specific applications, convergence can be accelerated by replacing our generic gradient estimator with an estimator that exploits the structure of the specific problem (e.g., parameterized regression).





Congestion Game with $N = 8$ players and $R = 4$ resources. $\eta_t = \frac{0.2}{t^{0.8}}$, $\delta_t = \frac{0.2}{t^{0.1}}$, $M = 6$. The action $\boldsymbol{x}_n \in [0,1]^R$ is the amount of usage of each of the R resources. The utility function is $u_n(\boldsymbol{x}) = \sum_{r=1}^{R} x_{n,r} - \frac{1}{20}\sum_{r=1}^{R} x_{n,r}\left(\sum_{m=1}^{N} x_{m,r}\right)^2$. Players broadcast one reward value per turn, uniformly at random. G is constant and random with average degree 4.25.

Variance (R1): R1 is correct that the variance increases like $\frac{1}{\delta_t^2}$, but the step size η_t multiplying the (estimated) gradient overcomes that. This is why $p - q > 0.5$, for which, $\sum_t \eta_t^2/\delta_t^2 = \sum_t t^{-2(p-q)} < \infty$ is required.

The Perturbations (R1): are indeed being independently generated across players. The reason is that the perturbation of player n, $\boldsymbol{z}_n$, is only meant to estimate the gradient of the sum of rewards wrt $\boldsymbol{x}_n$. The gradient approximation argument is used N times independently, and no two perturbations ever serve the same purpose. Hence, for player $m \neq n$, $\boldsymbol{z}_n$ serves as undesired (but necessary) noise in their communicated reward value (so no expectation is 0).

Lipschitz Factor (R1): A computation very similar to that suggested by R1 is done in (45) to bound the bias from the M steps delay. L appears in almost all the variance and bias terms in our proofs, affecting the convergence time. However, the theorem guarantees convergence for all L, and η_t (or δ_t) do not have to be tuned with respect to L.

Ergodic Markov Graph (R2,R4): $\mathcal{S}_G$ is a set of states of the process $G(t)$, so it is a set of graphs. We now denote this set $\mathcal{G}$ to avoid confusion. The union is between graphs i.e., $\bigcup_{G\in\mathcal{S}_G} G$ as opposed to $\bigcup_{G\in\mathcal{S}_G}\{G\}$ (using standard notation), which we now explicitly mention. R2 is correct that $G(t)$ transitions into $G(t+1)$ with no restrictions except that it constitutes an ergodic Markov chain. It is also correct that the algorithm does not know the graph. The purpose of this Markovian model is to capture a general time-varying process. This is the reality with physical agents, that move in space or that objects in space move around them, changing the configuration of the network. We agree that this is not the most important part of the paper. A useful and simple special case of this model (**R4**) is indeed when G is constant in time and connected or strongly connected (if directed). While not many algorithms can work under this general and challenging model, we are not the first to consider this well established model (see [34]). We hope that our explanation and the new notation make clear that the formulation is indeed rigorous and precise.

M **versus** N **(R2):** We discuss the requirement on M in Section 7.1 in the appendix. For a constant graph G, M needs to be larger than the diameter of G, and then R2 is correct that $M \geq N$ is always enough (and does not hurt much). However, for some Markov Chain (MC) $G(t)$, $M = N$ might not be enough if, for example, player 1 is isolated most of the turns and has to wait to get connection (edges). Hence, rigorously, a large enough M exists if the MC is ergodic, which might be an unknown function of the stationary distribution of $G(t)$. We now clarify this point in the theorem statement, and provide a simple example to illustrate this. We do not claim that M **needs** to be smaller than N. However, typically $diameter(G) \ll N$ ($O(\log N)$ for a random Erdős–Rényi G), so the memory complexity is low. We also claimed that $\mathcal{U}_n^M(t)$ can be much smaller than $\mathcal{N}$, meaning that at turn t, player n knows much less than the rewards of all players. This can happen even with $M = N$, if players broadcast only one reward value.

Known Actions (R4): Two practical examples where actions are never revealed: **(1)** Multi-agent Deep RL: like in GANs, the actions are the weights of a DNN that determine the "physical actions" of the agent (e.g. an autonomous vehicle), and the reward is a designed performance score over period of time. **(2)** Wireless networks: the action is the transmission power P_m, and device n measures the interference $\sum_m g_{m,n}P_m$ and cannot measure an individual action P_m. Nevertheless, our algorithm is appealing even with observable neighbors' actions, since it's unclear how one can efficiently use this information. First, each player needs to collect the entire action profile $\boldsymbol{x}_t$ from all over the network, for each t. A random subset will not do since $\sum_n u_n(\boldsymbol{x}_t)$ is non-linear in $\boldsymbol{x}_t$ (but it is in $u_n(\boldsymbol{x}_t)$, which is why our algorithms works). $\boldsymbol{x}_t$ are of higher dimensionality than the rewards so the communication overhead is larger.

Convergence Time Analysis (R2,R3): is highly complicated and requires novel techniques and proofs. Hence, we believe it should be pursued in a different paper. Our numerical experiments give some indication that the convergence time is reasonable, and behaves like SGD (the complicated part is the effect of G and the communication protocol).

Minor Comments (R1,R2): We resolved all minor comments (e.g., typos, using "optimal" more carefully).