Asynchronous vs Synchronous Selfish Learning

October 15, 2018

1 Introduction

In order to decrease the temporal variability experienced by WNs applying online learning, we propose the following mechanism:

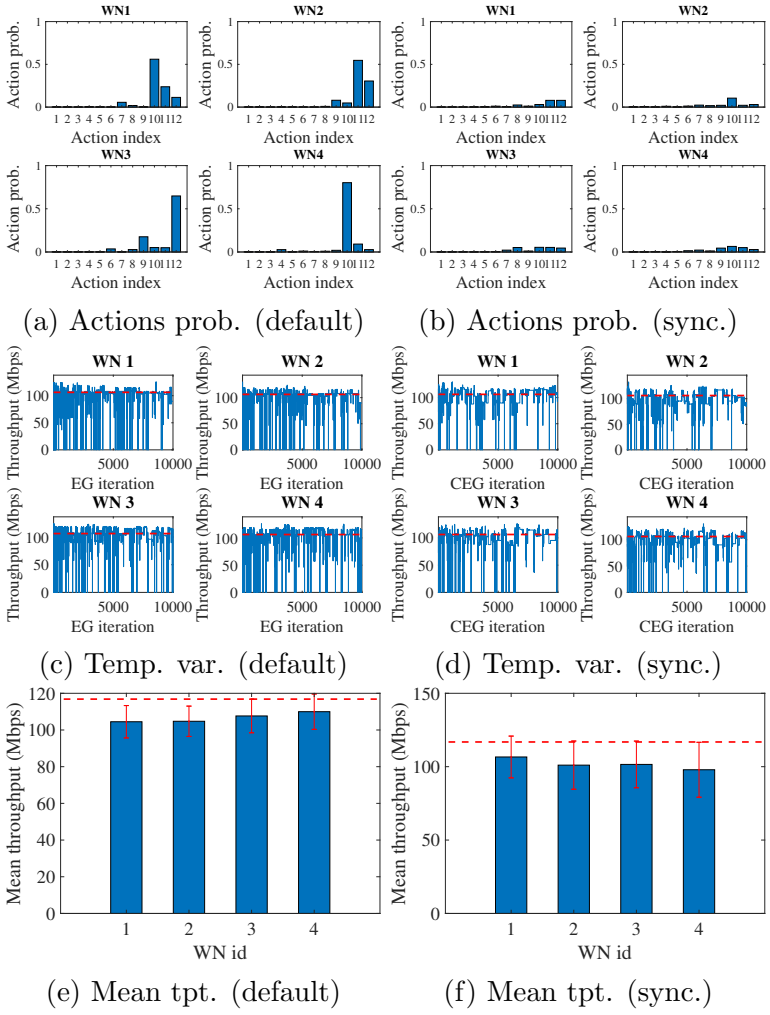
- WNs are synchronized, so that they act at specific intervals.
- In each iteration, only one WN is able to select an action. Of course, it can select the one that is currently using (exploitation).
- Every WN that remains "static" in a given iteration is considered to choose the last selected action.

Two important implications are derived from the abovementioned mechanism: *i)* the action-selection strategies of the learning algorithms are modified, *ii)* the adversarial setting changes, so that the performance of every chosen action is potentially evaluated in a larger set of situations. According to the latter, a higher level of valuable knowledge is more likely to be provided to each arm. However, depending on the learning algorithm used, there can be other implications.

2 Results - Toy Scenario

2.1 ε-greedy

Figure 1 shows several results regarding the application of ε-greedy in the toy scenario (1,000 iterations are considered), both for the fully decentralized and the synchronized approaches. As shown, the latter allows to experience a lower temporal throughput variability (1(c) vs **??**), since a lower number of actions is being exploited (**??** vs **??**). However, similar results are obtained with respect to the average throughput experienced per WN (1(e) vs **??**). The fact is that, for the synchronized approach, WNs tend to exploit sub-optimal actions for longer periods (while they wait for their turn). In contrast, WNs are able to rapidly discard these sub-optimal actions in their turn.



(a) Actions prob. (default) (b) Actions prob. (sync.)

(c) Temp. var. (default) (d) Temp. var. (sync.)

(e) Mean tpt. (default) (f) Mean tpt. (sync.)

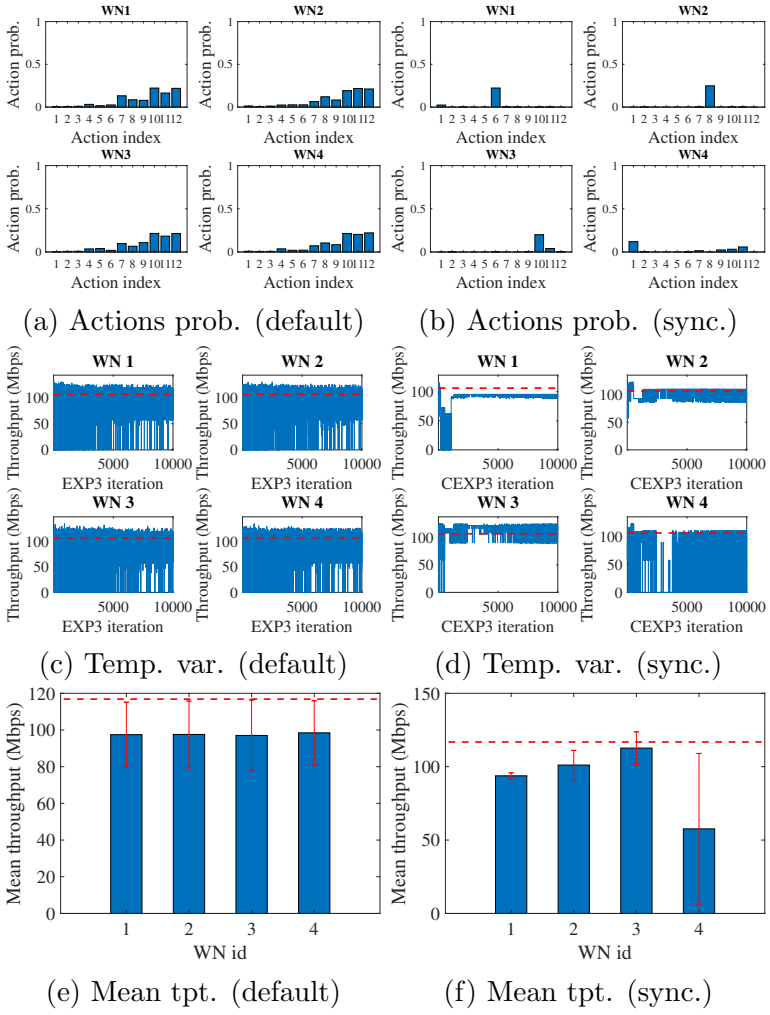
Figure 1: Simulation results in ε-greedy (1,000 iterations)

2.2 EXP3

Figure 2 shows several results regarding the application of EXP3 in the toy scenario (1,000 iterations are considered), both for the fully decentralized and the synchronized approaches. Again, the synchronized version of EXP3 leads to a lower temporal variability (2(c) vs **??**). No significant conclusions can be drawn for the actions probability profile (**??** vs **??**), since in EXP3 it appears to be slightly random (recall that different EXP3 simulations may lead to different results). Finally, regarding the average throughput experienced per WN, similar "pseudo-randomized" results are obtained (2(e) vs **??**).

2.3 UCB

Figure 3 shows several results regarding the application of UCB in the toy scenario (1,000 iterations are considered), both for the fully decentralized and the synchronized approaches.



(a) Actions prob. (default) (b) Actions prob. (sync.)

(c) Temp. var. (default) (d) Temp. var. (sync.)

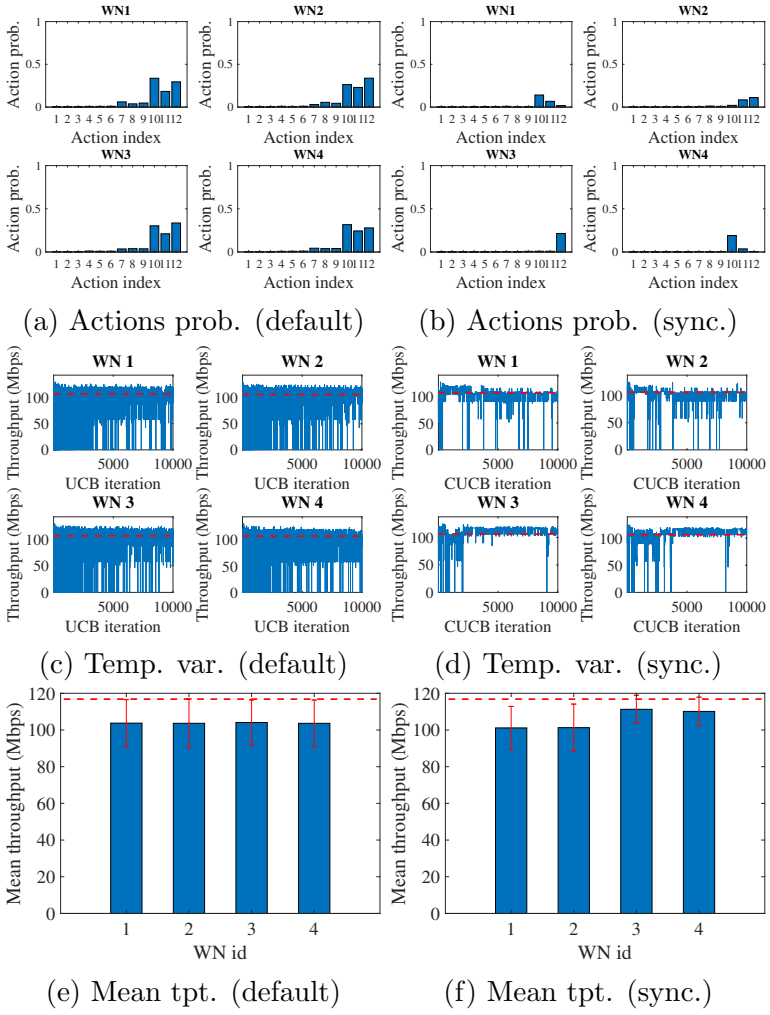
(e) Mean tpt. (default) (f) Mean tpt. (sync.)

Figure 2: Simulation results in EXP3 (1,000 iterations)

Unlike the previous cases, applying the synchronized approach to UCB is counter-productive, since its operational mode is altered by forcing WNs to choose undesired actions. As a result, the action-selection strategy is somehow randomized.

2.4 Thompson sampling

Figure 5 shows several results regarding the application of Thompson sampling in the toy scenario (1,000 iterations are considered), both for the fully decentralized and the synchronized approaches. In this case, the temporal variability is significantly decreased (4(c) vs **??**). In contrast, the actions probability changes for the synchronized approach. In this case, some WNs (2 and 3) alternate two between actions. As it can be inferred from the temporal variability, one of the actions (the one granting lower performance) is selected until iteration 600 (approximately). At that moment, the best performing action (with respect to the



(a) Actions prob. (default) (b) Actions prob. (sync.)

(c) Temp. var. (default) (d) Temp. var. (sync.)

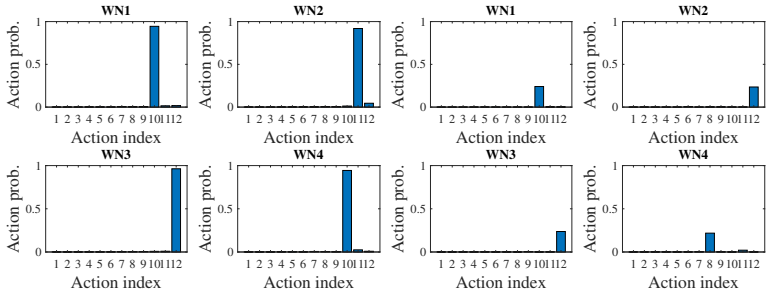
(e) Mean tpt. (default) (f) Mean tpt. (sync.)

Figure 3: Simulation results in UCB (1,000 iterations)

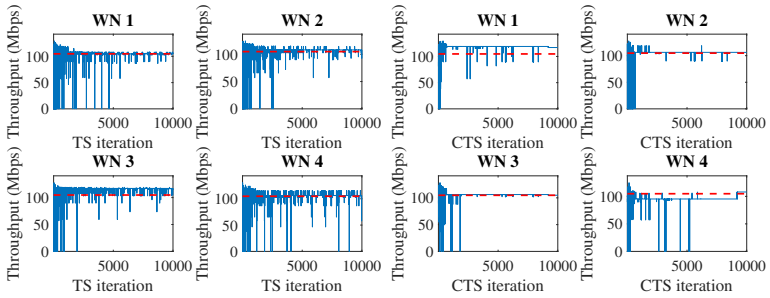
adversarial environment) is chosen. This phenomena allegedly occurs when near-to-optimal actions are forced to be selected several times during the "static" periods. As a result, the algorithm provides high estimates to that actions, which are hard to be overtaken by the actual best-performing ones.

3 Learning with penalties

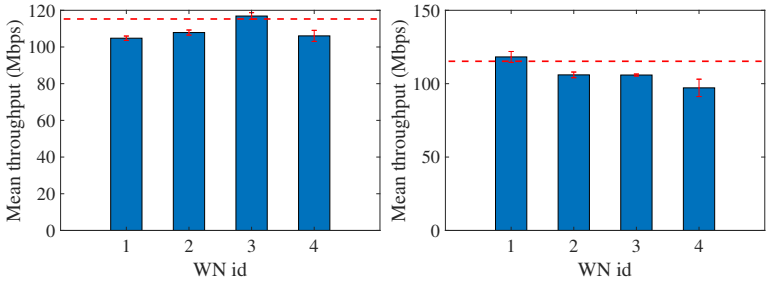
One promising possibility is to explore the effect of applying penalties to the reward, based on the transmit power or the saturation of a given channel (for that, we must know the number of users in each channel). Such penalties are not trivial to be derived, and when the problem becomes non-convex, the complexity increases. For that, we can provide some evidences about their usefulness when applying pure decentralized learning, but left its proper analysis as future work.



(a) Actions prob. (default) (b) Actions prob. (sync.)



(c) Temp. var. (default) (d) Temp. var. (sync.)



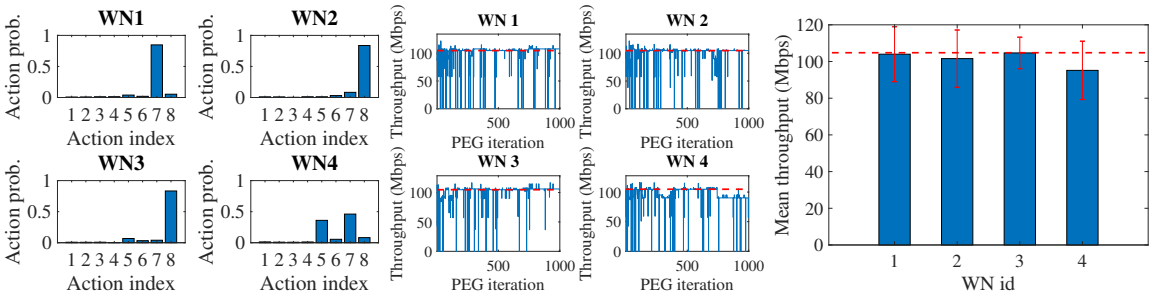
(e) Mean tpt. (default) (f) Mean tpt. (sync.)

Figure 4: Simulation results in Thompson sampling (1,000 iterations)

Here are some results for e-greedy when applying a penalty ($\alpha = 0.1$) to the transmit power used. The reward is now computed as:

$$R = \frac{\Gamma}{\Gamma^*} - \alpha\left(\frac{TxPower}{max(TxPower)}\right) \tag{1}$$

As shown, WNs focus on few actions, which grant highest performance by considering the trade-off between the throughput and the power transmitted.



(a) Actions prob. (penalty) (b) Temp. var. (penalty) (c) Mean tpt. (penalty)

Figure 5: Simulation results in Thompson sampling (1,000 iterations)