Collaboration and Competition Between Bandits:

Sharing Resources in a Constrained Environment

By Liam Purvis, Alexandre Brenelliere, Cheng-Hsiang Ho*, & Yilun Tang**

Abstract:

Multi-armed bandits are increasingly being applied to a variety of applications. However, due to the rare and unique kinds of data they operate on, we still don't have good intuition on how they behave in common environments that violate some of their key assumptions. In this paper our team creates an radio network environment that simulates which bandits transmit the most information across a discrete set of frequencies in a shared environment, one that penalizes collisions and switching channels. We sought to establish which combinations of algorithms yielded the best results, both individually and collectively, as well as visualize the patterns these bandits fall into. We compared these results against random and industry standard (CSMA) benchmarks. Our results indicate that UCB is typically the best choice for all metrics, (individual and collective), save for a few collaborative metrics that Thompson excels in. Additionally, we found that Thompson adapts poorly to the penalty for switching channel, and would perform much better with certain adjustments for this specific environment.

Contents:

*Not in EECS 227C, but collaborators on project which also served as MEng Capstone

Approach

Multi-armed Bandits

A finite set of discrete frequencies, each with its own probability of successfully sending a message, is a scenario that multi-armed bandits were designed to optimize. While standard supervised Machine Learning assumes we have perfect information in hindsight after making our predictions, information we can optimize our loss functions off of, Multi-armed Bandit algorithms are typically used when we can only see the loss for the specific action we took, and the best choice we could have taken is not clear in hindsight. Bandits are designed to reduce our posterior regret, as compared to the posterior regret of the ideal algorithms, by balancing the explore vs exploit tradeoff (Gittins et al, 2011).

Technical Literature Review

There are numerous algorithms for multi-armed bandits and many papers describing them. These algorithms include Upper Confidence Bound (UCB) (Auer, 2000), Thompson Sampling (Russo et al, 2017), Contextual Bandits (Beygelzimer et al, 2011), and more. Some variants exist for UCB and Thompson Sampling in non-stationary environments. (Garivier et al, 2008) However, due to the fact that these algorithms operate on data sets or environments that are rarely publically available (e.g. user recommendation click-throughs, suggested friends on social media, or radio environments) there is still not a good understanding of which algorithms are typically more successful and, even more interesting, how various permutations of these algorithms collaborate in a zero-sum game. While machine learning has had many instances of exploring which algorithms work best, both on real data and on a simulated dataset (Scikit-learn, n.d.), there is an opportunity to attempt similar discovery in the multi-armed bandit field. We aim to explore this by simulating a radio network and forcing algorithms to collaborate on it as their actions interfere with each other.

Methods and Materials

Design of Radio Environment

Model of signal loss

Our simulation environment had to make several decisions on how to best model radio communications. First, we selected a simplified loss model which determines signal strength based on transmitter power output and the squared distance between the listening point and the transmitter's position. We did not factor in other components, such as the effect of frequencies on signal loss, since the experiments are being run with small scale frequency distribution, and such a factor is not significant at this level.

Second, we focused down the spectrum bandwidth to be between 1000 MHz and 1100 MHz. After inspection with common design (WiFi is at 2.4 GHz), we determined that this range of

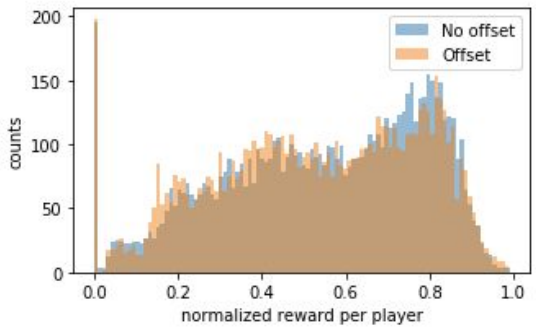
frequencies is most popularly used, especially among smaller sized and mobile devices, which we deemed most relevant for our project due to their popularity and wide-spread usages.

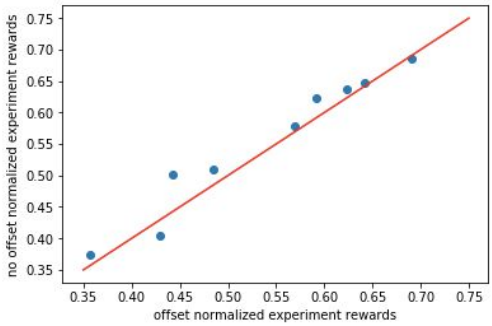
Definition of success and Signal-to-Noise Ratio

We also use a simplified Signal-to-Noise model to determine whether certain messages are successfully delivered or not. Within our simulation environment, we define specific Signal-to-Noise ratios (SNRs) that every transmitter-receiver pairs must abide to. The signals strength is determined by the transmitter's power level and the distance in between, while the noise strength is determined by other transmitter-receiver pairs' communication, treated as noises. If the SNR calculated is beyond a specified threshold, then the information is completely lost.

Time-steps and why time references have little influence

Our environment also performs simulations on a time-step basis, each step representing 0.1s. Doing so allows us to simulate radio communications in a more manageable manner. Because of this design choice, our simulation environment will evaluate at discrete pre-defined time steps. In order to diversify and mimic how real world communication does not always operate on and same clock or sharing a same reference of timing, the simulation environment itself runs on a finer scale of time steps than each individual transmitter - receiver pair does.
We wanted to check if having different time references for different radios significantly affects the results. To do this, we ran a small subset of experiments with and without time offsets.





Based off these figures, we concluded that there is no significant difference between both cases. Since having different time references has a high impact on computational performance (X10 running time), we decided to get rid of them for our final set of simulations.

Penalty for switching channel

Each time a player switches to a different channel, the player will be frozen from transmitting for the next 4 time steps to simulate the cost of updating in the real world. This setup poses a challenge for our bandit algorithms, as they are not designed to take such a penalty into account. Implementing such mechanics provides additional realism to our simulations, as we imagine physical equipment needs time to change settings, and with this penalty factor we can imitate the applied performances of our algorithms better.

Ignoring some real-life features such as error correction

In our current design, our transmitter-receiver pairs share the same definitions of channels. Such design does not always occur in real world, and our simulation environment is capable of making it more realistic, but we made this choice for computational reasons.
Without the coding and error correction, the success of transmission only depends on the SNR of the channel, which makes the performance of the algorithm only depend on its ability to find available resources. For the design of our simulation environment, we choose not to consider the factor of coding/decoding and error correction for more efficient performance and more concentrated simulation factors.

Description of algorithms

*UCB***:** Balances explore and exploit by choosing channel with maximum reward plus confidence interval.
Thompson: Uses bayesian updates to randomly choose channels, according to the probability they are the best.
Discounted: Each Bandit had a variant which added decreasing weights to past rewards, to handle non-static environments *CSMA:* Commonly-used industry standard. Uses randomized timesteps to listen to and send information.
Random: Benchmark for performance comparisons.

Selection of algorithms

We selected 6 staple algorithms to perform simulations on, with consideration from the commonality and popularity perspectives. There are Random, Carrier Sense Multiple Access (CSMA), UCB, and Thompson sampling, and their variants UCB discounted and Thompson discounted. We use the Random type behaviors as benchmark for the other more algorithmic approaches, a standard approach in Machine Learning research topics. The Random type transmitter-receiver pair would change channels randomly based on a binomial random variable, given a predetermined paramter. Similarly, we selected CSMA because it is a common approach for collision avoidance in radio communications. Factoring in CSMA in our simulations allows us to compare other bandits algorithm with what is considered as the industry standard.

Experimentation setup

We identified two main environment types:

- Static: we call “static” an environment where channel capacity are constant over time
- Non-static: we call “non-static” an environment where channel capacity can change over time (because of changing noise or of some incumbent periodically changing its bandwidth use)

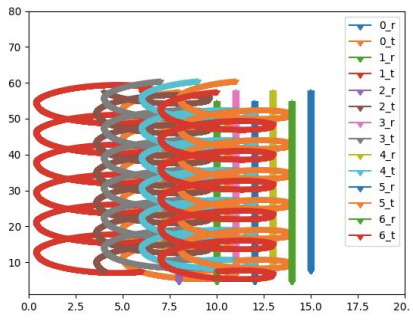
These two types of environments must be tested in three situations, each usually leading to different behavior for common algorithms:

- Unconstrained resources: there is more bandwidth available than required for all players to transmit successfully
- Exact resources: there is exactly as much bandwidth as needed for all players to be successful
- Constrained: there isn’t enough bandwidth for all players to transmit simultaneously

In addition, we test the effectiveness of each algorithm when they have different proportion of players in simulations. The four kinds of proportion being tested are:

- Single: there is only one type of player in the simulation.
- Majority: most of the players in the simulation are using the observed algorithm.
- Minority: only a few of the players in the simulation are using the observed algorithm.
- Even: exactly the same amount of the players in the simulation using the observed algorithm

We also run our simulations in mobile and motionless situations. Motion patterns are directly inspired from DARPA SC2 challenge to be as realistic as possible.



Run-time environment

We ran all of our experiments in a Ubuntu 18.04 virtual machine running on Google cloud. We were using a 4-core CPU.
Our running times per simulation are :

- For 1500 timesteps when we had a static environment: 1s (non-mobile) / 5s (mobile)
- For 5000 timesteps when we had a non-static environment: 4s (non-mobile) / 19s (mobile)

Data Analysis

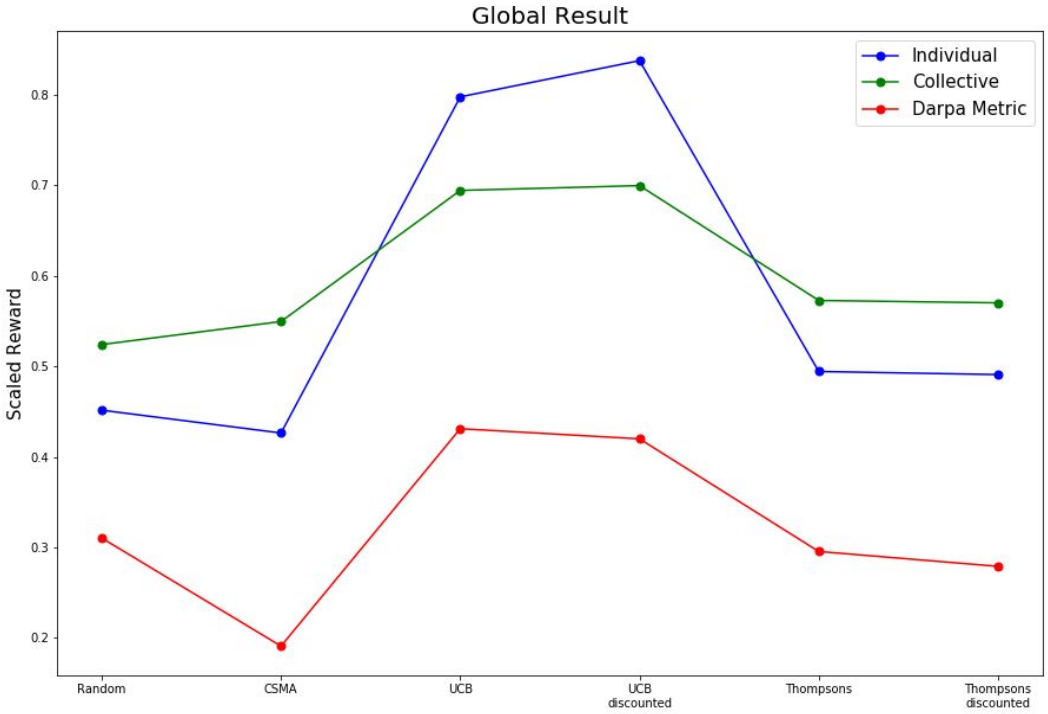
Overall result

We first measured the performance of each algorithm against three metrics:

- Individual score: this is the cumulated success of the player, scaled to the number of steps
- Collective score: this is the sum of the cumulated successes of all players in the simulation, scaled to the number of steps and players
- Darpa score: the darpa score is trying to balance individual and collective performance in one metric (it is inspired of the DARPA SC2 challenge). It rewards the protection of the weakest players.
  If any player scored less individually than a threshold, then all players get the individual score of the worst player as darpa score. If all players were better than the threshold, then they all get their own individual score as darpa score.

First Findings:

1. Generally, a player that performs well individually will perform well for the other metrics.
2. UCB is significantly better than the others for all metrics.
3. Thompson is barely better than random when it comes to darpa metric.



Interpretation

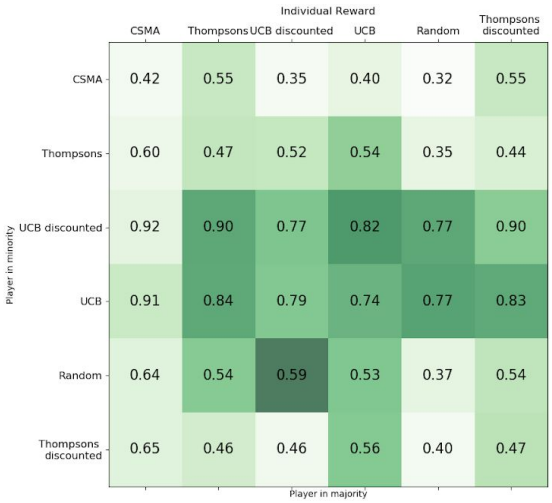
1. In most cases, not all resources are used by the players : there are always some unused channels. A good player is mostly a player able to find the empty channels, so it's not harming the other players.
2. By design, UCB is very good at finding empty channels and is stable. This helps a lot for getting a high score.
3. Random naturally protects weak players since there is only a small probability to be conflicting with any of the other players. So we would expect Random to get a good darpa score.

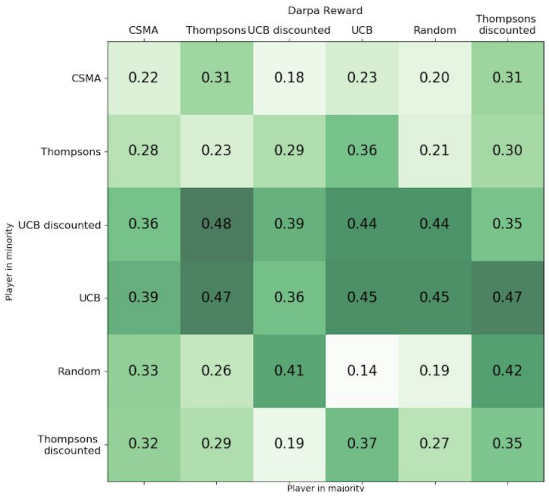
Compatibility matrices

We wanted to understand which pairs of algorithms perform well in a minority-majority situation. We designed the compatibility matrix for this purpose:

- Columns = player in majority
- Rows = player in minority
- Value display = score of the player in minority

(Note: The 95% confidence bound of the results is virtually always within ± 0.01, and was calculated via bootstrapping)





Findings

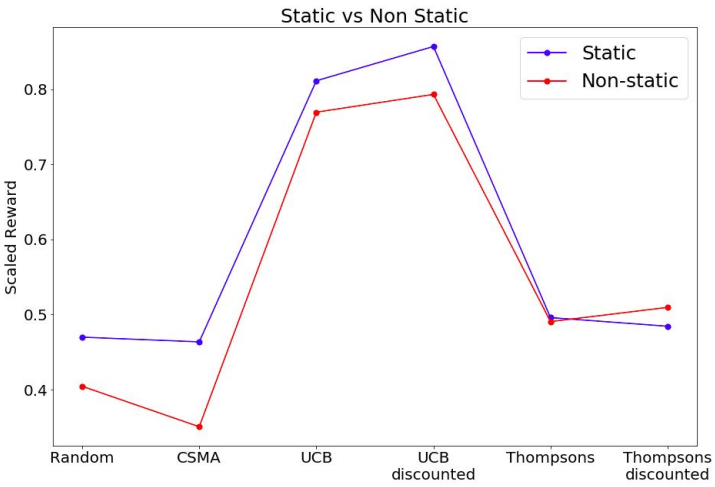
1. UCB is the best algorithm in almost all categories. It is particularly excellent at individual score.
2. The difference between UCB and Thompson is much smaller for the darpa score, compared to individual score.

Interpretation

1. Same as before, UCB is very well suited for our experiments and gets consistently good results across situations.
2. Thompson is much better at protecting weak players and distributing bandwidth. We expect that reducing its instability could make it get better darpa scores than UCB.

How non-stationarity affects results

Since we had players that are supposed to be specifically suited for stationary (UCB and Thompson) and non-stationary environments (UCB discounted and Thompson discounted), we decided to benchmark all of the algorithms in both types of environments.
A non-static environment is when the capacity of channels changes over time because of external factors.



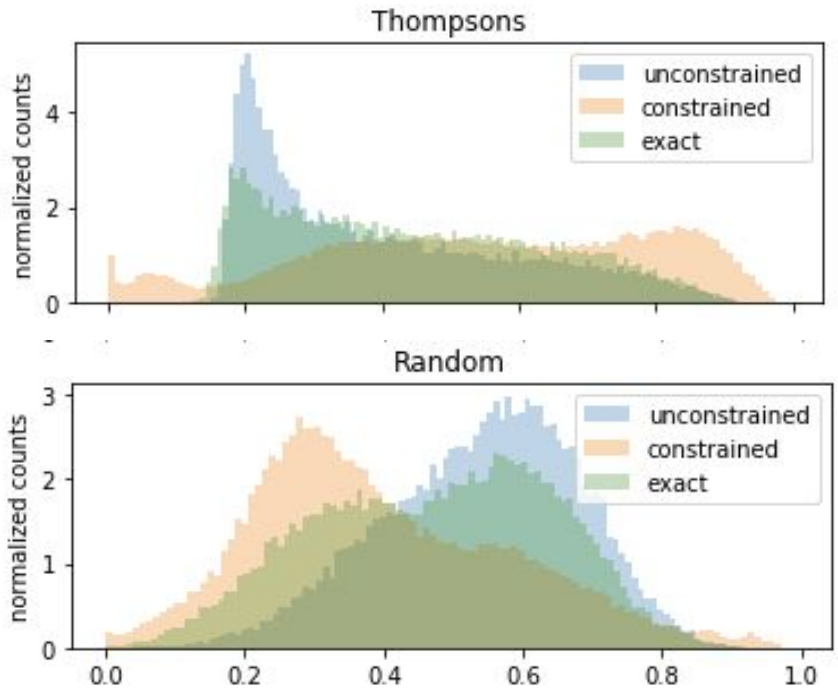
Findings

There is no difference in the hierarchy of players between static and non-static environments, except for a slight difference for Thompson discounted.

Interpretation

1. All of our environments were actually non-stationary: even if there was no external factor influencing the channels' capacities, the high number of players, each of them switching channels, automatically creates a non-stationary environment. So it makes sense that we don't see much difference in the results.

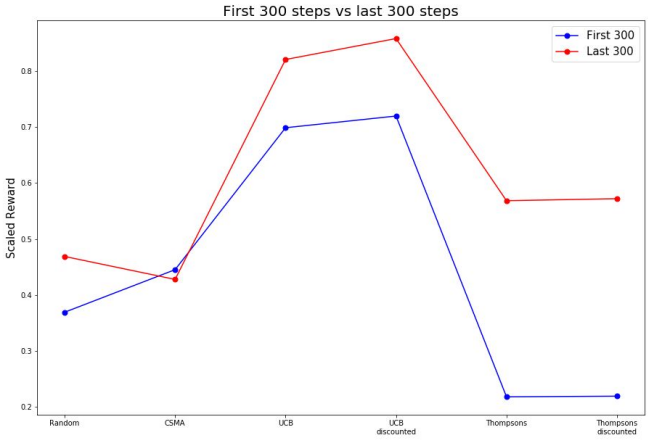
Thompson suitability for radio-environments:



Here, we can see that Thompson actually does better in a constrained vs unconstrained environment. This is counterintuitive, until we remember that Thompson Sampling doesn't take into account the penalty for switching channels. As such, in an open environment where there are many different channels with equally good rewards, it will switch across many channels continually, often racking up unnecessary penalties. This plot implies that if we tweaked the fundamental algorithm behind Thompson Sampling to avoid incurring this loss, it might perform at a much higher level.

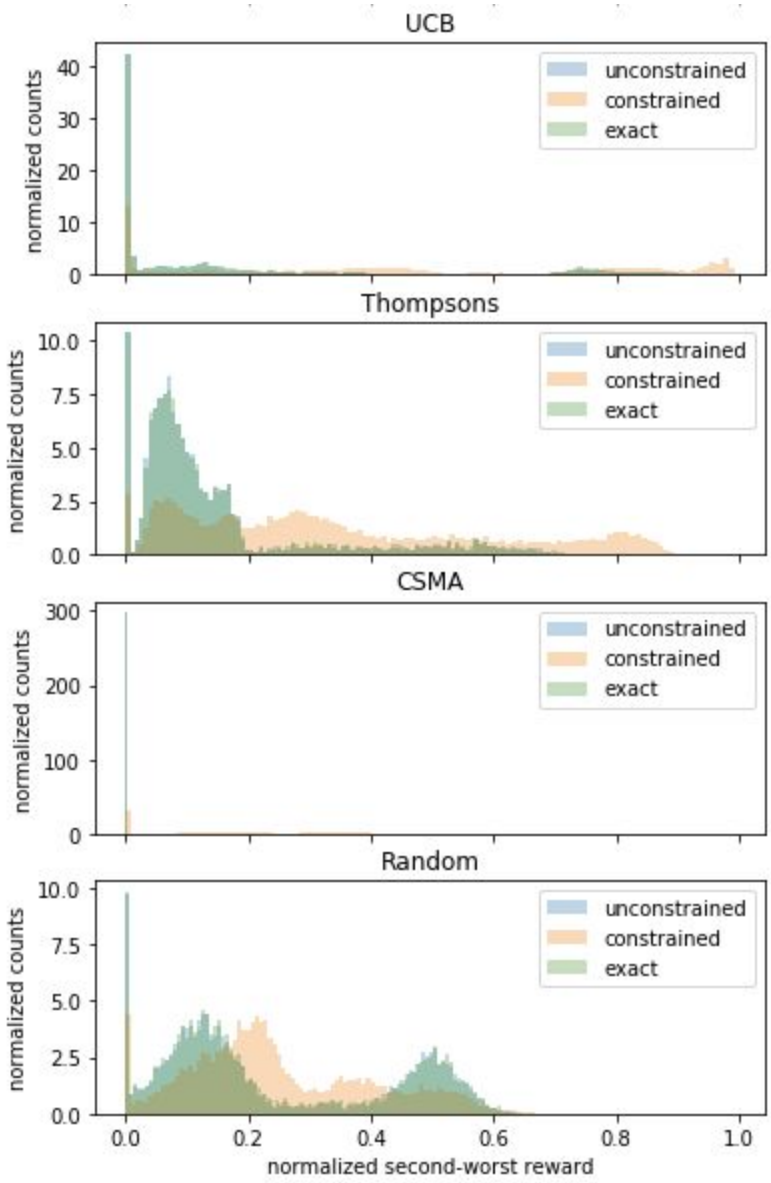
Thompson needs time to adapt

Considering the performance of the Thompson is sometimes only slightly better than the Random and CSMA, we explore further details on the Thompson behavior. The figure below shows the scaled reward of



the first 300 steps and the last 300 steps of each type of player.

Thompson Sampling Helps Lowest Performers more than UCB:



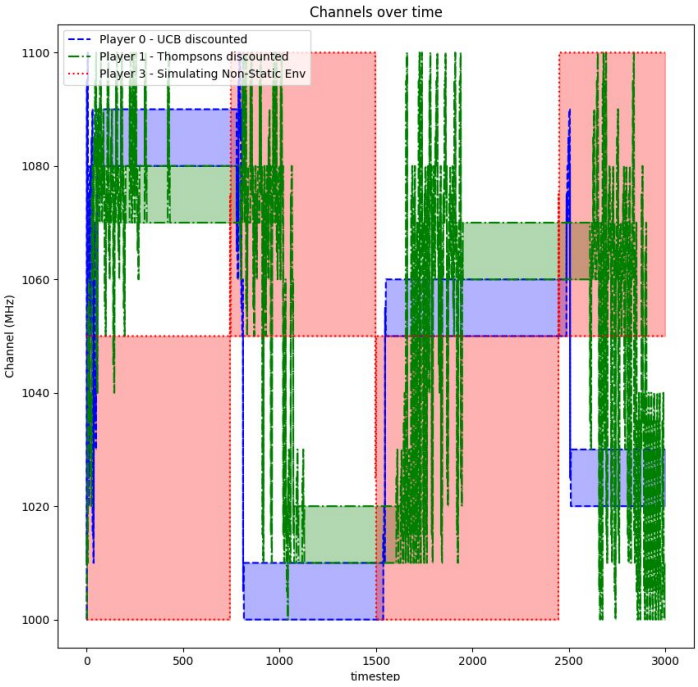
Here, we can see the effect that the use of each has on the second lowest scoring player in the environment. We can see that UCB is more polarizing, that in general it seems Thompson performs better than UCB, that CSMA might be the worst for this metric and Random might be the best.

Findings

1. It can be said that both UCB and Thompson perform worse at the beginning compared to the later stage, while the CSMA and Random have the almost the same rewards no matter which stage they are in.

2. The difference of the Thompson Sampling is even more obvious than the UCB players. The result explains why the overall reward of Thompson sometimes is only slightly better than the random and CSMA, which is due to the poor performance at the beginning of the experiments.

Interpretation



1. The low reward for UCB and Thompson are caused by the high switching penalty for frequently exploring channels in the beginning, which is not cost in Random and CSMA. While at the later stage, the searching process becomes shorter as there is more information collected, and therefore the performance of UCB and Thompson become better than Random and CSMA as they are able to adapt to the new change fast.
2. In addition, by observing the detailed behavior of UCB and Thompson as shown in the figure above, it is shown that Thompson takes more steps to find a new channel when the current channel is no longer available. The difference explains why the gap between the reward of first 300 and last steps in Thompson is larger than the one in UCB.

Conclusion

Using both individual and collective performances, we established evaluation metrics that compare two popular multi-armed bandits algorithms (UCB and Thompson sampling) with other benchmark algorithms, and synthesized their successes with a variety of visualizations. The results demonstrate that UCB and UCB Discounted consistently dominate in almost every metric. Far behind it lie Thompson, CSMA, and our random benchmark. Thompson's behavior suggests it is poorly suited to the penalty of switching channels, and could potentially be modified to account for the context of its actions. When examining metrics that take into account the collaboration of these different algorithms, Thompson's does do almost as good as or better than UCB in certain respects. These results imply that any single algorithm should not be taken as uniform across all metrics, but UCB will likely be your first choice if you were starting blind to the dynamics of your environment.

References

Auer P. “Using upper confidence bounds for online learning”. *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 270–279. IEEE. 2000. https://ieeexplore.ieee.org/document/892116

Beygelzimer, Alina, Langford, John, Li, Lihong, Reyzin, Lev, and Schapire, Robert E. “Contextual Bandit Algorithms with Supervised Learning Guarantees”. *Journal of Machine Learning Research*. Volume 15. 2011

O’Shea, Tim. West, Nathan. “Radio Machine Learning Dataset Generation with GNU Radio”. *6th GNU Radio Conference*. 2016

“Classifier comparison”. Scikit-learn. Scikit-learn. Accessed 2019/02/20. https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Gittins, John C., Glazebrook, Kevin D., and Weber, Richard. *Multi-Armed Bandit Allocation Indices*. 2nd ed. Chichester: Wiley, 2011.

Aurélien Garivier, Eric Moulines. *On Upper-Confidence Bound Policies for Non-Stationary Bandit.* Problems. 24 pages. 2008.

“Internet of Things Forecast”. Mobility Report. Ericsson. Accessed 2019 Feb 20. https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

Reardon, Marguerite. “Post 9/11: Can we count on cell networks?”. *CBS News*. Last Modified: 2011 Sept 7th. Accessed: 2019 Feb. https://www.cbsnews.com/news/post-9-11-can-we-count-on-cell-networks/

Russo, Daniel, Van Roy, Benjamin, Kazerouni, Abbas, Osband, Ian, and Wen, Zheng. “A Tutorial on Thompson Sampling”. 2017 July 7.

“The Technology”. Spectrum Collaboration Challenge. DARPA. Accessed 2019 Feb 20. https://www.spectrumcollaborationchallenge.com/about/technology/