Achieving Proportional Fairness in WiFi Networks via Bandit Convex Optimization

Golshan Famitafreshi, Cristina Cano

Achieving Proportional Fairness in WiFi Networks via Bandit Convex Optimization*

Golshan Famitafreshi and Cristina Cano

Universitat Oberta de Catalunya, Castelldefels Barcelona 08860, Spain
gfamitafreshi@uoc.edu
ccanobs@uoc.edu

Abstract. We revisit in this paper proportional fair channel allocation in IEEE 802.11 networks. Instead of following traditional approaches based on explicit solution of the optimization problem or iterative solvers, we investigate the use of a bandit convex optimization algorithm. We propose an algorithm which is able to learn the optimal slot transmission probability only by monitoring the throughput of the network. We have evaluated this algorithm both using the true value of the function to optimize, as well as adding estimation errors coming from a network simulator. By means of the proposed algorithm, we provide extensive experimental results which illustrate the sensitivity of the algorithm to different learning parameters and noisy estimates. We believe this is a practical solution to improve the performance of wireless networks that does not require inferring network parameters.

Keywords: Bandit Convex Optimization · Proportional Fairness · WiFi.

1 Introduction

Bandit Convex Optimization (BCO) is a type of Online Convex Optimization (OCO) in which we deal with partial information. In BCO, decisions are made between a player and an adversary repeatedly. In each iteration, the player selects a point from a fixed and known convex set. Then the adversary chooses a convex cost function. At the end of the iteration, the only available feedback for the player is the cost of the function at the selected point. In this framework, the player does not have any knowledge about the specific function nor the gradient [6]. The main emphasis of BCO in the machine learning community has been on rigorous theoretical performance analysis of algorithms. However, practical application of BCO algorithms still requires more attention.

We argue that since many wireless network optimization problems can be easily formulated as convex problems; BCO is appealing for the wireless networking community. Some potential applications of the convex optimization formulation

* This research is partially funded by the SPOTS project (RTI2018-095438-A-I00) funded by the Spanish Ministry of Science, Innovation and Universities.

are pulse shaping filter design, transmit beamforming, network resource allocation, MMSE precoder design for multi-access communication, robust beamforming and optimal linear decentralized estimation.

BCO has two important advantages in wireless network communication. The first advantage is that the player only needs the cost function feedback of a given action, which facilitates practical implementation. Second, in BCO, the adversary is able to choose among a set of convex functions which can capture network dynamics such as changes in the number of nodes and channel conditions.

The main contribution of this article is an investigation of how to apply a bandit convex optimization algorithm to proportional fair resource allocation in wireless networks. This approach can be implemented by the access point allowing learning of the optimal slot transmission probability only by monitoring the throughput of the network. This research can help academia as well as practitioners to assess whether bandit convex optimization algorithms can be a practical solution for commercial use. The ultimate goal is to bring optimal channel allocation in WiFi to practice by addressing the limitations of traditional approaches that need to be fed with as well as track changes in network parameters (such as packet size and data rate used by each node in the network).

This paper is organized as follows. Section 2 describes the main background: the random back-off operation and proportional fair allocation analysis in WiFi networks. Section 3 describes bandit convex optimization and the proposed algorithm. Section 4 presents the evaluation setup and performance results. Section 5 summarizes the related work in the area of WiFi network proportional fair optimization. Finally, in Section 6 some final remarks are given.

2 IEEE 802.11 Background

In this section first, we describe the random back-off operation, then summarize the throughput optimization in WiFi networks based on proportional fairness.

2.1 Random Back-off Operation

The IEEE 802.11 protocol employs the Distributed Coordination Function (DCF) mechanism to access the channel, which is basically a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) method with binary exponential back-off. In DCF when a station wants to send a packet, it monitors the channel. If it senses the channel idle for a distributed inter-frame space (DIFS), it will start a back-off countdown timer. Each time the station starts the back-off procedure it initializes CW to $\mathrm{CW}_{\min}$ and chooses a random number in $(0, \mathrm{CW} - 1)$, where CW is the contention window. As long as the channel is sensed idle for a time slot the back-off timer counter decrements. When a transmission is detected on the channel, this timer freezes and is reactivated when the channel is sensed idle for more than DIFS again.

2.2 Throughput Optimization in WiFi Networks

Throughput in the 802.11 standard depends on the number of active stations and the contention window used by each station. In particular, in multi-rate IEEE 802.11 WLANs, stations that use DCF and transmit at lower transmission rates make use of the channel for longer periods of time to transmit the same amount of data compared to stations using higher transmission rates. This reduces the throughput of high-rate stations in the WLAN, since less time is available for transmission in the shared medium. This effect is known as performance anomaly. One solution to approach this problem is proportional fair allocation [10].

As done in [5] we formulate proportional fairness as a convex optimization problem whose objective is the sum of logs of throughput, which will intrinsically capture fairness while trying to achieve maximum performance even in a multi-rate scenario. We assume that all the stations (n) are saturated, (i.e. stations always have a packet to send) and generate uplink traffic to the network. Note that the slot transmission probability (τ) is the reciprocal of the value of CW. Let $S_i(\tau)$ be the throughput of the station i, then:

$$S_i(\tau) = \frac{P_{\text{succ},i} D_i}{\sigma P_{\text{idle}} + T_c(1 - P_{\text{idle}})}. \tag{1}$$

Here two kinds of time slots are considered. The first one is the PHY idle slot duration without any transmission which is of duration σ. The second one is the busy slot which relates to the duration of a packet transmission [7]. The packet transmission duration is denoted by T_c, which is the mean duration of a successful or collided transmission of node i or other stations packet transmissions. The successful transmission, includes the MAC ACK which is specified by T_{ack}, T_{fra} which defines the duration of a data transmission, a short inter-frame space (SIFS) which represents an amount of time that channel requires for sending the response frame and a DIFS. T_c can be calculated as follows (for clarity of illustration, we consider this duration equal for successful transmissions and collisions but the analysis can be easily extended to consider both as done in [8]):

$$T_c = T_{\text{fra}} + \text{SIFS} + T_{\text{ack}} + \text{DIFS}. \tag{2}$$

The average packet size of the i_{th} station is defined by D_i in bits. $P_{\text{succ},i}$ is the probability of a successful packet transmission of i_{th} station, i.e., that only station i transmits a packet and is given by $P_{\text{succ},i} = \tau_i \prod_{k=1,k\neq i}^{n}(1 - \tau_k)$. The term P_{idle} is the probability that the channel is idle. When none of the stations attempt to transmit a packet, the probability is defined as $P_{\text{idle}} = \prod_{k=1}^{n}(1 - \tau_k)$. The term $1 - P_{\text{idle}}$ is the probability that the channel is busy due to the successful, unsuccessful (collisions) or other stations packet transmissions and it is defined by $1 - P_{\text{idle}} = 1 - \prod_{k=1}^{n}(1 - \tau_k)$. Therefore, Eq. 1 is the amount of data transmitted per slot when that is successful over the average duration of a slot.

In the following it will be more useful to use the transformed variable $x_i = \frac{\tau_i}{(1-\tau_i)}$ rather than τ_i, $x_i \in [0, \infty)$ for $\tau_i \in [0, 1)$. The optimization problem can then be formulated as [5]:

$$\begin{aligned} \text{max.} \quad & \sum_{i=1}^{n} \tilde{S}_i(x), \\ s.t. \quad & \tilde{S}_i(x) \leq \log \frac{x_i D_i}{X(x) T_c}. \end{aligned} \tag{3}$$

The constraint certifies that the sum of logs of throughputs is feasible and sits in the rate region. Since the log-transformed rate region $\tilde{Z}$ is strictly convex, there exists a unique solution that satisfies strong duality and Karush-Kuhn-Tucker (KKT) conditions which implies a global maximum [5].

This problem can be solved explicitly but our aim here is to formulate this problem in a bandit framework so that knowing the networks parameters in Eq. 3 is no longer required. This facilitates practical implementation and can help adoption of optimization approaches in commercial WiFi cards, where the selection of the CW used by the stations is generally static.

3 Bandit Convex Optimization

In Bandit Convex Optimization (BCO) three steps are repeated between the player and the adversary. These three steps can be written for iteration t as follows:

- The player chooses a point $x_t \in K \subseteq R^d$.
- The adversary chooses a cost function $f_t \in F \subseteq R^k$.
- The player observes $f_t(x_t)$.

Here, x_t is a point from a fixed and known convex set. K represents a convex subset of a d-dimensional Euclidean space ($K \subseteq R^d$).In addition, all the functions in F are convex [4].

The aim of BCO algorithms is to achieve low regret:

$$R_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x). \tag{4}$$

This formulation is known as cumulative regret, which measures the difference between the cumulative loss that is revealed to the player and the best-fixed decision in hindsight after T iterations [6]. To achieve low regret most of the BCO algorithms use Online Gradient Descent (OGD) with estimations of the gradient. In fact, the main complication of BCO is to estimate the gradients of the cost functions. Therefore many researchers in BCO have investigated methods for estimating these gradients and used their results in BCO algorithms [6].

Flaxman et al. [2] proposed a scheme that combined the estimated gradients with the OGD algorithm of Zinkevich [11], who showed that a simple gradient descent strategy for the player incurs a $O(\sqrt{T})$ regret bound [3]. The algorithm of Flaxman et al. uses point evaluations of convex functions to approximately estimate the gradient. The regret bound of this algorithm is shown to be $O(T^{3/4})$.

Agarwal et al. showed that in each round knowing the value of each cost function at two points is almost as useful as knowing the value of each function everywhere, therefore their algorithm has a regret bound of $O(T^{2/3})$ improving the $O(T^{3/4})$ bounds achieved by Flaxman et al. However, Flaxman et al. and Agarwal et al. approaches cannot be used in a practical implementation and realistic setting in wireless networks. First, in many settings it is impossible to query cost functions two times in one iteration. Second, the variance of the single point estimators in the approach of Flaxman et al. [2] is large; consequently, speed of convergence is not practical for wireless stations [4][9].

3.1 Sequential Multi-Point Gradient Estimates in WiFi

We use the multi-point BCO algorithm defined in [4] which considers a simpler assumption than that of Agarwal [3]. This algorithm is called Online Gradient Descent with Sequential Multi-Point Gradient Estimates (OGD-SEMP) and to estimate the gradient queries are combined from two consecutive iterations. The algorithm considers a sequence of auxiliary points $y_1, y_2, ...$ which are used to keep track of the player's movement by updating gradient descent as follows:

$$y_{k+1} = \prod_{k}(y_k - \eta_k \tilde{g_k}). \tag{5}$$

Here, $\tilde{g_k}$ is the gradient estimate which is used to update y_{k+1}. The parameter η_k is the gradient descent step size and $\eta = \{\eta_1, \eta_2, \eta_3, ...\}$ is a sequence which shrinks over time. This coefficient defines the speed of convergence of gradient descent to the final value y_k. Figure 1 shows a schematic of this algorithm for the k_{th} iteration. Let's consider y_k as the k_{th} point in a one-dimensional convex set with the interval $[y_k - \epsilon_k\delta_k, y_k + \epsilon_k\delta_k]$. Then, the distance between the beginning and the end of the interval is $2\epsilon_k\delta_k$. Here, ϵ_k is a random number that can be either -1 or 1. δ_k is a parameter which shrinks over time and it captures the distance between the selected point and y_k. In the first step, we choose an arbitrary point and obtain its cost function as:

$$\begin{aligned} \bar{x_t} &= y_k + \epsilon_k\delta_k, \\ {g_k}^+ &= f_t(\bar{x}_t). \end{aligned} \tag{6}$$

In the second step we choose another point in time $(t+1)$ and obtain its cost function as:

$$\begin{aligned} \bar{x}_{t+1} &= y_k - \epsilon_k\delta_k, \\ {g_k}^- &= f_{t+1}(\bar{x}_{t+1}). \end{aligned} \tag{7}$$

In the following step, the gradient can be estimated as follows:

$$\tilde{g_k} = \frac{{g_k}^+ - {g_k}^-}{2\epsilon_k\delta_k}. \tag{8}$$

Here, the numerator is the subtraction of two cost functions evaluations and the denominator is the distance between the beginning and the end of the interval (see Figure 1), which corresponds to an unbiased estimator of the gradient.

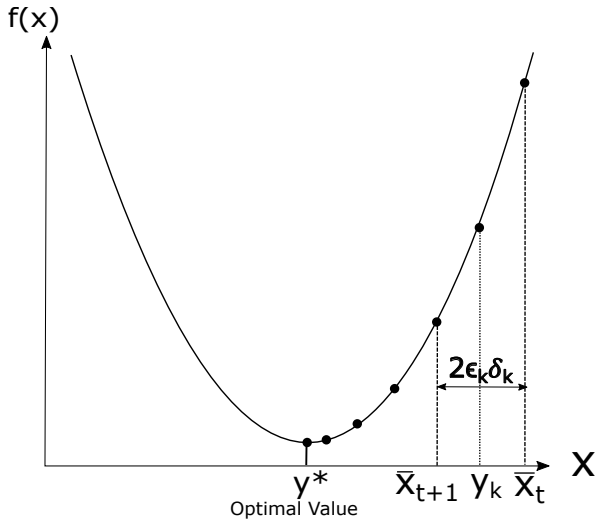


Fig. 1. Sketch of Online Gradient Descent with Sequential Multi-Point Gradient Estimates.

This algorithm was used in [4] for wireless networking optimization. The authors provided the theoretical analysis of this algorithm and evaluated its performance in an unlicensed LTE/WiFi fair coexistence use case. Here we aim to use this method for achieving proportional fair allocation of resources in a IEEE 802.11. Using this method, we only need to know the throughput of each station, regardless of other parameters of Eq. 3, to achieve proportional fairness.

Applied to the WiFi throughput proportional fairness case, the cost function is equal to $f_t = \sum_{i=1}^{n} \tilde{S}_i(x_t)$. Therefore, the cost functions in Eq. 6 and Eq. 7 ($f_t(\bar{x}_t)$, $f_{t+1}(\bar{x}_{t+1})$) define the objective function of our optimization problem at time t and time $t+1$ respectively, with $\bar{x}_t = \{x_1, x_2, ..., x_i\}$ a vector which defines x_i for all the stations at time t,[1] and $\bar{x}_{t+1} = \{x_1, x_2, ..., x_i\}$ a vector which defines point x for all the stations at time $t+1$.

In more detail, consider a repeated game of T rounds. In each round $t = \{1, 2, ..., T\}$ the WiFi network selects Eq. 3, which we now denote as f_t with some n, x_i, D_i and T_c. Then, in each round t:

[1] Recall that x_i is a transformed variable $x_i = \frac{\tau_i}{(1-\tau_i)}$ rather than τ_i, $x_i \in [0, \infty)$ for $\tau_i \in [0, 1)$. (as seen in section 2).

- The WiFi access point chooses $\bar{x}_t$.
- The WiFi network (formed by all WiFi nodes) independently selects $f_t \in F$.
- The WiFi access point observes $f_t(\bar{x}_t)$.

4 Performance Evaluation

Here we present the evaluation of OGD-SEMP when applied to the WiFi use case by executing an extensive set of simulations in Matlab and in a custom simulator. According to the IEEE 802.11ac standard, we have considered:

$$T_{\text{fra}} = T_{\text{plcp}} + \left\lceil \frac{L_{\text{s}} + n_{\text{agg}}(L_{\text{del}} + L_{\text{mac-h}} + D) + L_{\text{t}}}{n_{\text{sym}}} \right\rceil T_{\text{sym}}, \tag{9}$$

$$T_{\text{ack}} = T_{\text{plcp}} + \left\lceil \frac{L_{\text{s}} + L_{\text{ack}} + L_{\text{t}}}{n_{\text{sym}}} \right\rceil T_{\text{sym}}, \tag{10}$$

with physical and channel access parameters as listed in Table 1.

This custom simulator consists of two main parts, the channel and the node module. The node module includes a network model with physical, MAC, network and application layers. It was previously used for evaluating OGD-SEMP performance in an unlicensed LTE/WiFi fair coexistence use case. In this research we extend the simulator to consider the particularities of the WiFi proportional fair use case gradient descent implementation.

Table 1. Simulation parameters according to IEEE 802.11ac [1].

Parameter	**Value**
Slot Duration (σ)	9 μs
DIFS	34 μs
SIFS	16 μs
PLCP Preamble + Header Duration (T_{plcp})	40 μs
EIFS	364 μs
TimerACK	314 μs
Propagation Time	1 μs
Tsymbol (T_{sym})	4 μs
PLCP Service Field (L_s)	16 bits
MAC Delimiter Field (L_{del})	32 bits
MAC Header ($L_{\text{mac-h}}$)	288 bits
Tail Bits (L_t)	6 bits
ACK Length (L_{ack})	256 bits
Payload (D)	12000 bits
nsymbol (n_{sym})	1040 bits
number of aggregated packets (n_{agg})	64
MIMO	4

Since the contention window values are discrete while the slot transmission probabilities in our model are continuous, we need to convert discrete values of contention window to the desired continuous one in the simulator. To achieve this we use CW_1 for t_1 seconds and CW_2 for t_2 seconds in a way that the average contention window matches that of the model.[2] Recall that $\tau = 1/CW$. The gradient descent algorithm is executed each T seconds, with $T = c(t_1 + t_2)$, with c a positive integer. The throughput used by the algorithm as feedback is the average throughput during T seconds.

We evaluate the performance of the algorithm using the individual throughput metric (S_t). We use Matlab in order to feed the algorithm with the true values of the individual throughput computed using Eq. 3 for each set of experiments and use the simulator for evaluating the impact of having estimated values instead of the true value of the throughput. Noise in the estimates is caused by the random backoff, collision probabilities and discretization of the slot transmission probability as described above. By comparing the evolution of throughput over time for different settings we evaluate the algorithm's performance regarding the time to convergence.

4.1 Simulation Results

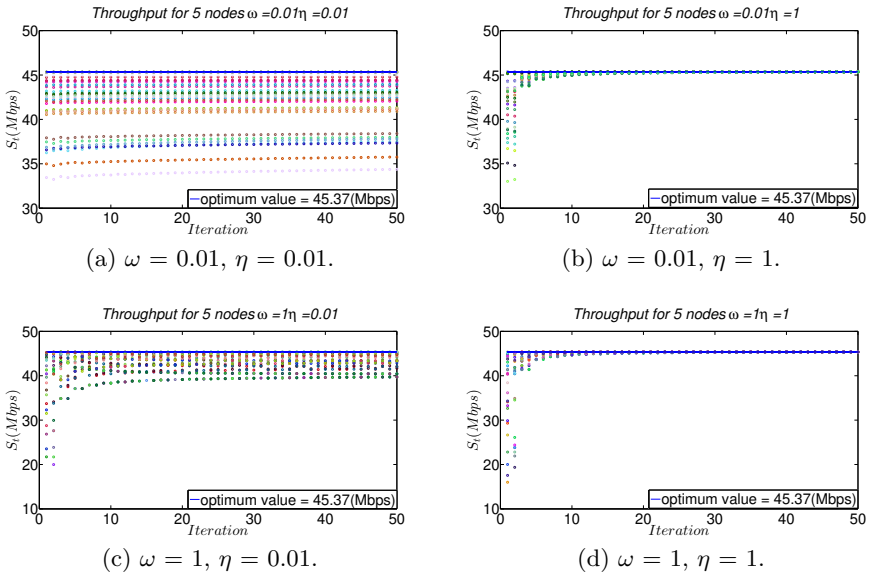
In the simulations first, we have set the gradient descent step size as $\eta_k = \eta/k^{(3/4)}$ and $\delta_k = \omega/h(k)$, with ω as an input parameter and $h(k)$ as some increasing function. We will refer to ω as the exploration parameter and h as the exploration schedule. We vary the number of stations in the network (n) with $n = \{5, 20\}$. Then in order to evaluate the sensitivity of the algorithm to two different exploration schedules, we have changed the exploration schedule to $\eta_k = \eta/k^{(1/2)}$. Note that stations always have a packet to transmit (nodes are saturated) and we consider homogeneous stations (same packet size and transmission probability).

Sensitivity to the Learning Parameters First, we evaluate the performance of the algorithm by changing the exploration parameter ω -observe that this parameter controls how far from y_k we take the two cost function evaluations at consecutive iterations- and gradient descent step size (η_k). We set $h(k)$ equal to $k^{(3/4)}$ which shrinks the exploration parameter as time goes by.

Figure 2 shows the results of the individual throughput for 5 nodes during 50 iterations. We repeat the same simulations for 30 runs in order to obtain more accurate results. Therefore, in the figures each color represents one simulation run. Optimal results from [5] are shown in Figure 2 as straight lines. This results are obtained from the implementation of the algorithm in Matlab with cost function computed using Eq. 3 and IEEE 802.11ac parameters from [1]. We show results with different exploration parameter $\omega = \{0.01, 1\}$ and gradient descent step size $\eta = \{0.01, 1\}$.

[2] In particular, we set CW_1 and CW_2 to the immediately lower/higher allowed value in IEEE 802.11 and compute t_1 and t_2 accordingly.

Figure 2 shows that by fixing η and increasing parameter ω the rate of convergence increases. As we saw in Eq. 6 increasing the value of exploration parameter (ω), we take bigger steps towards the optimum. By fixing parameter ω and increasing the parameter η for the range of values considered the rate of convergence increases as well since we also make larger steps with bigger gradient descent step sizes. We observe that the increasing trend in the second case is faster than the first case. It can be seen that for exploration parameter equal to $\{0.01, 1\}$ and gradient descent step size equal to 1 the algorithm converges to the optimum value after a few number of iterations (less than 20).

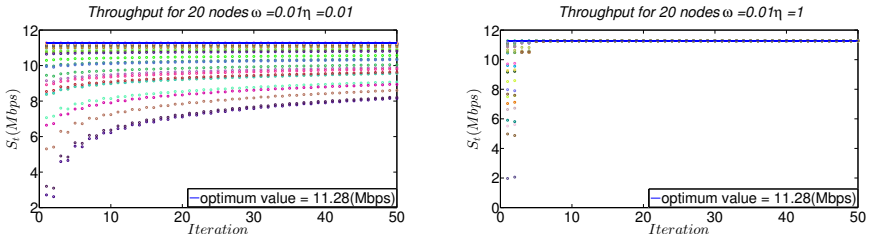


(a) $\omega = 0.01$, $\eta = 0.01$. (b) $\omega = 0.01$, $\eta = 1$.

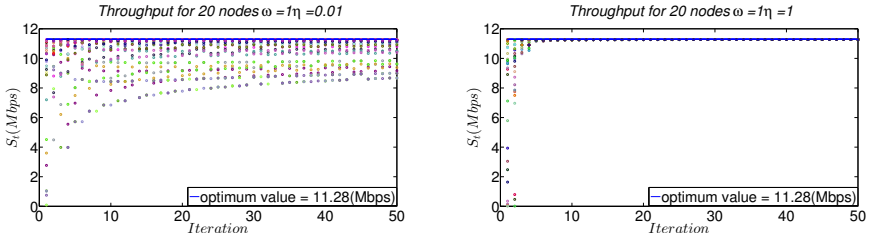
(c) $\omega = 1$, $\eta = 0.01$. (d) $\omega = 1$, $\eta = 1$.

Fig. 2. Individual throughput for $n = 5$ using different ω, η and $h(k) = k^{(3/4)}$.

Here we keep the algorithm setup same as above and increase the number of nodes. Figure 3 illustrates the individual throughput of a network 20 nodes. By comparing Figure 2 and 3, we observe that for the same value of ω and η by increasing the number of nodes, the algorithm converges to the optimum faster. The reason for this behavior is illustrated in Figure 4. This figure shows the shape of the objective function for the different number of nodes. As it is shown by increasing the number of the nodes in the network the function becomes steeper, thus the gradients are larger. This means that the algorithm makes larger steps at each iteration and reaches the optimum value faster. Note that the difference between the minimum value of the convex function and its maximum is increased by increasing the number of nodes. For this case the algorithm converges in around 10 iterations or less for $\omega = \{0.01, 1\}$ and $\eta = 1$.

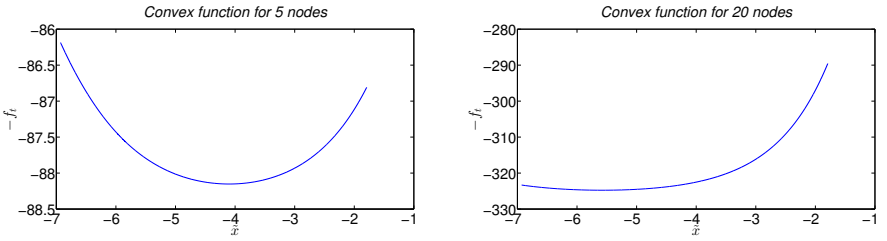


(a) $\omega = 0.01$, $\eta = 0.01$. (b) $\omega = 0.01$, $\eta = 1$.



(c) $\omega = 1$, $\eta = 0.01$. (d) $\omega = 1$, $\eta = 1$.

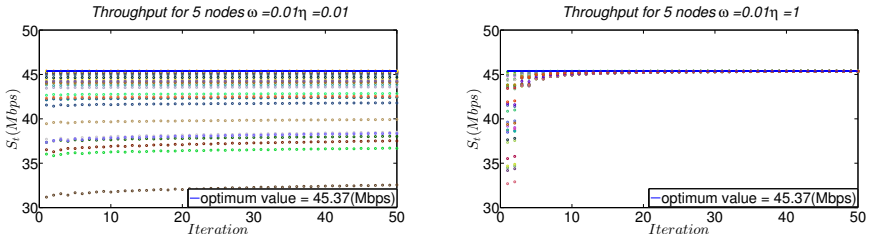
Fig. 3. Individual throughput for $n = 20$ using different ω, η and $h(k) = k^{(3/4)}$.



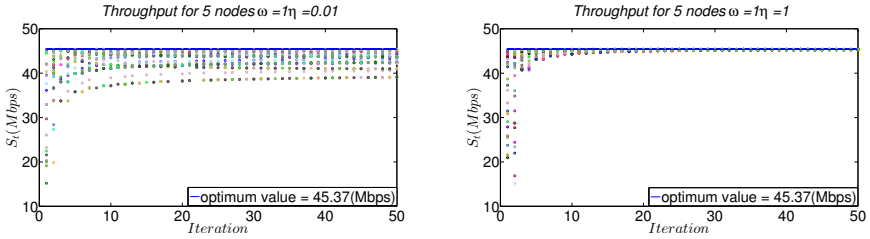
(a) $n = 5$. (b) $n = 20$.

Fig. 4. Shape of the objective functions for $n = 5$ and $n = 20$.

Sensitivity to the Exploration Schedules In this set of simulations we use the same setup as in the previous subsection but we change the exploration schedule to $h(k) = k^{(1/2)}$. Here we evaluate the sensitivity of the algorithm to two different exploration schedules. Similarly to Figure 2 and Figure 3, Figure 5 and Figure 6 show the individual throughput for different values of ω, η and $h(k) = k^{(1/2)}$. We can observe that with $h(k) = k^{(1/2)}$, the convergence speed is almost the same as $h(k) = k^{(3/4)}$. Only for the case $n = 5$, $\omega = 1$ and $\eta = 1$ (Figure 2(d) and Figure 5(d)), we observe that the convergence speed in Figure 2(d) with $h(k) = k^{(3/4)}$ is slightly faster than in Figure 5(d) with $h(k) = k^{(1/2)}$. These results show that the sensitivity of the algorithm to the exploration schedule is negligible for the exploration schedules considered.

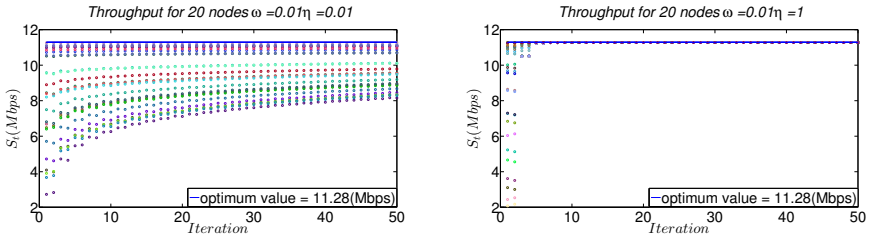


(a) $\omega = 0.01$, $\eta = 0.01$. (b) $\omega = 0.01$, $\eta = 1$.

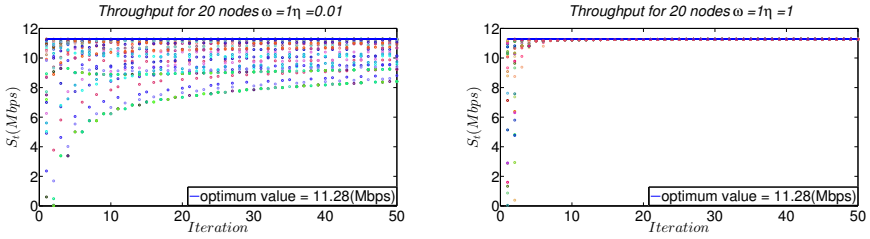


(c) $\omega = 1$, $\eta = 0.01$. (d) $\omega = 1$, $\eta = 1$.

Fig. 5. Individual throughput for $n = 5$ using different ω, η and $h(k) = k^{(1/2)}$.



(a) $\omega = 0.01$, $\eta = 0.01$. (b) $\omega = 0.01$, $\eta = 1$.



(c) $\omega = 1$, $\eta = 0.01$. (d) $\omega = 1$, $\eta = 1$.

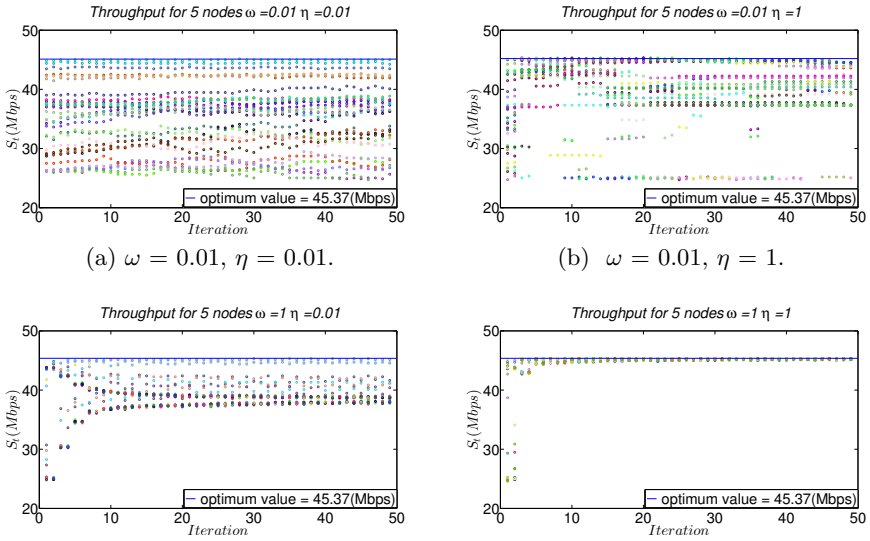
Fig. 6. Individual throughput for $n = 20$ using different ω, η and $h(k) = k^{(1/2)}$.

This means that the impact of $h(k)$ compared to the gradient descent step size η, exploration parameter ω and gradients is low. We also see, as before, that

by increasing the number of nodes in the network the individual throughput value converges to its optimum value faster (See Figure 6).

Sensitivity to Noisy Gradient Estimates Here we evaluate the performance of the algorithm by having noisy estimates of the individual throughput instead of the true value. In order to achieve this goal we implement the algorithm in the simulator. We set the exploration parameter to $\omega = \{0.01, 1\}$ and gradient descent step size to $\eta = \{0.01, 1\}$. We set the gradient descent timer equal to $T = 100$ s and the value of contention window timer equal to $(t_1 + t_2) = 0.1$ s. Each simulation is again run 30 different times in order to achieve more accurate results. Here the exploration schedule is set to $k^{(3/4)}$.

Figure 7 shows that the algorithm still converges in less than 10 iterations for $\omega = 1$ and $\eta = 1$. We see that for $\omega = 0.01$ the evolution of throughput is not following the desired convergence trend. We also see that convergence is worse for smaller values of the exploration parameter. The reason for this behavior is, we argue, the noise: with a small value of ω the gradient estimations are less accurate making more probable for gradient descent to move in the opposite direction of the optimum.



(a) $\omega = 0.01$, $\eta = 0.01$. (b) $\omega = 0.01$, $\eta = 1$.

(c) $\omega = 1$, $\eta = 0.01$. (d) $\omega = 1$, $\eta = 1$.

Fig. 7. Individual throughput for $n = 5$ using different ω, η and $h(k) = k^{(3/4)}$ (noisy estimates).

5 Related Work

The most recent works on WiFi network throughput optimization are based on a proportional fairness approach. The reader is referred to those which are presented by Checco et al. [5] and Patras et al. [8], which are similar in nature.

Checco et al. [5] pioneered rigorous analysis of proportional fairness in IEEE 802.11 WLANs. They proved that a unique proportional fair rate allocation exists as the flow total air-time. This algorithm corrects previous works on air-time quantities and uses the IEEE 802.11 rate region as a log-convex. It satisfies per station fairness and per flow fairness. In these approaches [5, 8], throughput optimization is achieved by inferring MAC parameters and network metrics such as packet transmission duration, slot transmission probability and average packet size of the stations. These metrics can be estimated but we need to handle estimation errors and network dynamics.

We base our algorithm on these rigorous approaches but without the need to know all parameters of the function to optimize. In this way proportional fairness can be achieved without the need to infer and keep track of network parameters and only by estimating the individual throughput at each station, which can be achieved at the application layer in a commercial access point with minimal changes.

6 Conclusion

The main focus of this article is on achieving proportional fairness in WiFi networks by applying bandit convex optimization. We have applied the OGD-SEMP algorithm based on the BCO algorithm to the WiFi proportional fairness use case. Our results show that, with the appropriate setting of parameters, the algorithm converges to the optimum value in a few number of iterations. However, the parameter of the algorithm that controls the degree of exploration has a significant impact on the algorithm's performance, especially when we are faced with throughput estimation errors. This can be alleviated by increasing the duration of the estimation periods but at the cost of longer convergence times. Other solutions involve using averages of the gradient estimates in different iterations. The evaluation of these approaches to reduce the sensitivity of the algorithm to noise are left as future work. We conclude that the algorithm is a practical solution for wireless network optimization, but that care has to be taken when configuring the algorithm parameters.

References

1. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4: Enhancements for Very High Throughput for Operation in Bands Below 6 GHz. ANSI/IEEE Standard 802.11ac (2013)
2. A. D. Flaxman, A.T.K., McMahan, H.B.: Online Convex Optimization in the Bandit Setting: Gradient Descent Without a Gradient. In: Proceedings of the Sixteenth

Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 385–394. SODA '05, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2005), http://dl.acm.org/citation.cfm?id=1070432.1070486
3. Agarwal, A., Dekel, O., Xiao, L.: Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback. In: COLT. pp. 28–40. Citeseer (2010)
4. Cano, C., Neu, G.: Wireless Optimisation via Convex Bandits: Unlicensed LTE/WiFi Coexistence. In: Proceedings of the 2018 Workshop on Network Meets AI & ML. pp. 41–47. NetAI'18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3229543.3229551, http://doi.acm.org/10.1145/3229543.3229551
5. Checco, A., Leith, D.J.: Proportional Fairness in 802.11 Wireless LANs. IEEE Communications Letters **15**(8), 807–809 (August 2011). https://doi.org/10.1109/LCOMM.2011.060811.110502
6. Hazan, E., et al.: Introduction to online convex optimization. Foundations and Trends® in Optimization **2**(3-4), 157–325 (2016)
7. Li Bin Jiang, Soung Chang Liew: Proportional fairness in wireless LANs and ad hoc networks. In: IEEE Wireless Communications and Networking Conference, 2005. vol. 3, pp. 1551–1556 Vol. 3 (March 2005). https://doi.org/10.1109/WCNC.2005.1424745
8. Patras, P., Garcia-Saavedra, A., Malone, D., Leith, D.J.: Rigorous and practical proportional-fair allocation for multi-rate Wi-Fi. Ad Hoc Networks **36**, 21 – 34 (2016). https://doi.org/https://doi.org/10.1016/j.adhoc.2015.06.002, http://www.sciencedirect.com/science/article/pii/S1570870515001262
9. Saha, A., Tewari, A.: Improved Regret Guarantees for Online Smooth Convex Optimization with Bandit Feedback. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 636–642 (2011)
10. Siris, V.A., Stamatakis, G.: Optimal CWmin Selection for Achieving Proportional Fairness in Multi-rate 802.11e WLANs: Test-bed Implementation and Evaluation. In: Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization. pp. 41–48. WiNTECH '06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1160987.1160996, http://doi.acm.org/10.1145/1160987.1160996
11. Zinkevich, M.: Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 928–936 (2003)