# Spectrum Adaptation in Multicarrier Interference Channels

Haleema Mehmood
Department of Electrical Engineering
Stanford University

## I. Introduction

Digital Subscriber Lines (DSL) are an example of multiuser interference channels (unless vectored). In a multiuser interference channel (IC), multiple users share a communication medium with no coordination on either the transmitting or receiving ends. Although DSL lines do not share the actual copper wire between users, the interference between adjacent lines in a single binder or cable effectively makes it an IC. Furthermore, DSL systems use Discrete MultiTone Modulation (DMT) which means that each user divides its bandwidth into multiple parallel, independent subcarriers or tones. This makes the system a muticarrier, multiuser interference channel [1].

Although DSL systems are increasingly moving towards central control and coordination of spectrum and bit allocation, traditionally, this has not been the case. A central system may dictate total data rate and sometimes power allocation at start-up, but each user or modem may be allowed to independently alter its bit and power distribution in response to changes in channel or noise characteristics during showtime (after system is ON and running). The purpose is to ensure that modems can maintain good operating margins in events of variation in channel and noise conditions. However, in the absence of central control and coordination, distributed adaptation can potentially cause system instability or long transient periods due to interference coupling between DSL lines.

Changes in DSL channel characterisitcs and noise and internal or external interference occur infrequently and stay the same over long durations of time. This gives DSL lines sufficient time to adapt and adjust their operating states according to the new conditions. The problem with adaptation however is: If one user changes its power allocation, it changes the interference it causes to other users. This in turn causes other users to change their own spectra as a response, causing a chain of spectrum adaptations done by each user individually, but affecting all users in the system because of the crosstalk couplings between the users' spectra.

This project aims to study if controlling the rate at which users adapt their spectra affects convergence behavior. The system is observed at discrete time steps. The rate of adaptation is defined as the number of subcarriers a modem is allowed to update during one time interval. The intuitive idea is that large, sudden changes in spectra by users may cause system instability. Smaller, slower changes may results in long transient periods, yet a more stable system. Given a particular system is there an optimal policy or some good policies that make the system go back to a stable state quickly. This project uses reinforcement learning on a multiuser, multicarrier, interference channel to

- study feasibility of allowing uncoordinated, independent adaptation of spectra by users.
- assess the effect of rate of bit and spectrum adaptation by individual users on system stability.
- find a control policy that ensures fast convergance to a desired stable state.

## II. System Background

In Discrete Multitone Modulation (DMT), each channel is divided into multiple subcarriers. Transmission over each subarrier depends upon the Signal to Noise Ratio (SNR) over that subcarrier. The number of bits that can be transmitted over a subcarrier $k$ by user $u$ is given by

$$b_{k,u} = log_2(1+p_{k,u}|H_{k,u,u}|^2/(\Gamma\lambda(\sum_{m=1 m\neq u}^{U}|H_{k,u,m}|^2 p_{k,m}+\sigma_{k,u}^2))).$$

Here $p_{k,u}$ is the transmit PSD of user $u$ on subcarrier $k$. $\sigma_{k,u}^2$ is the variance of the Gaussian background noise and $H_{k,u,m}$ is the channel gain on subcarrier $k$ from users $m$ to user $u$. The total number of users in the system is $U$ and the number of subcarriers for each user is $K$. $\Gamma$ is the SNR gap defined for a target bit error ratio, and $\lambda$ is the *target SNR margin*. The total data rate for each user $u$ is then $R_u = \sum_{k=1}^{K} b_{k,u}$, and the rate is achieved subject to a per-user power constraint $\sum_{k=1}^{K} p_{k,u} \leq P_t$.

As the equation shows, the bits and powers over all subcarriers are coupled because of the cross-channel couplings. A discrete bit-loading procedure, the Levin Campello (LC) algorithm can be run iteratively to assign initial bits and powers to all users on all subcarriers. Once the system is initialized, an incremental update procedure using bit swaps and gain adjustments, is employed for further adpatation. The LC algorithm calcualtes the entire spectrum and bit distribution. On the other hand, adaptive bit swap and gain adjustment procedures only vary the bit or power allocation on a few tones at a time.

The Iterative Levin Campello (ILC) algorithm has a target SNR margin $\lambda$ that each user should attain. The margin is the extra power on each subcarrier beyond that needed to support the bit allocation, provided as a safeguard against changes in channel and noise conditions. Usually a 6dB margin target is used in most DSL systems. While absolute convegence means that when the iterative algorithm concludes, each user should have 6dB margin on each of its tones, such convergence seems unattainable for discrete bit algorithms for most cases [2]. Therefore the definition of convergence needs to be relaxed. The criterion employed for this project is that each user should have a final margin within some tolerance of 6dB at the end of the procedure. Afterall, the margin is precisely meant to protect against changes in the channel and so is the adaptive update procedure. That is why we can allow the adaptive update procedure to use up some of the margin. For a real system, this is a very practical and safe assumption.

## III. MDP Formulation

From the previous section, it is clear that given a DSL system, the variables of interest are: $p_{k,u}$, $b_{k,u}$ and $\lambda_{k,u}$. Given the power and bit distributions, the channel coeffiecients and external noise can be used to calculate the per subcarrier margins. The system is observed in discrete time steps. It is assumed that each user has a target data rate and a total power constraint that has to be met at all times. The system is therefore always in a state where the bits add up to the target rate and the powers add up to satisfy the power constraint. The only thing that is not satisfactory are the subcarrier margins. Let $\delta$ be a threshold (in dB). If $|6-\lambda_{k,u}| < \delta \forall k, u$, then the system has attained a stable state and no user needs to update anything anymore. If the subcarrier margins are not within this limit for certain users on certain tones, then those users need to update their powers &/or bits on those tones to fix the subcarrier margins. At each time step, therefore, each of the users measures the noise and interfernce on its line, measures its subcarrier margins and updates its bit and power allocations on certain subcarriers. This is done in parallel by all users.

The problem above is formulated as a markov decision process (MDP). **Value iteration** is used to find the optimal strategy after forming the model. The strategy is tested by simulation and compared against other random and strategic policies. The parameters of interest are:

**States,** $S$: The state for each user is defined as the number of subcarriers with unacceptable margins. Notice that absolute margin values have not been used. Only a count of the number of tones with unacceptable margins is used. This was done to simplify the analysis. The per-user state is further quantized to 10 values. Table 1 shows what the quantized states represent.

**Actions,** $A$: The actions allowed to each user are defined by the maximum number of subcarriers that are allowed to be updated in one turn. 10 actions are allowed to each user. Table 1 shows what the actions represent.

For this project, a 2-user system is analyzed. For such a system, there are a total of $10^2$ states and actions. The states and actions follow similar partitioning of the space of subcarriers for each user as shown in Table III

**State Transition Probabilites,** $P_{sa}$: The state transition probablities are learned by running a 2-user simulation. By running the simulation many times, the 100x100x100 (s,s',a) matrix of transition probabilities is calculated.

Note that the DSL system itself is deterministic. It is the way the states and actions are defined (as number of subcarriers and not the actual bit/power/margin values on those subcarriers) that makes the state transitions probabilistic, depending upon actions.

**Reward,** $R$: The reward is a function of the state only, with 0 reward for the desired state which has all subcarriers satisfying the margin constraint. Once this state is reached, the system has converged. For all other states, the reward is $-1$, i.e. it is kept the same for all unacceptable states. Although different rewards can be given to different states based on how many subcarriers have unacceprable marings, but the focus of this project is on ensuring fast convergence. That is why, all bad states are given the same reward of $-1$. Also, it was not clear how giving different rewards was affecting the learning process, and this particular reward function was easier to work with than some of the other reward functions that were tried.

| S/A | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| NumSub | 0/2 | 4 | 8 | 16 | 32 |
| S/A | 5 | 6 | 7 | 8 | 9 |
| NumSub | 64 | 128 | 256 | 512 | 1174 |

TABLE I
PER-USER STATES/ACTIONS AND CORRESPONDING NUMBER OF BAD/TO-FIX SUBCARRIERS

## IV. SIMULATION

The model for the MDP is learned by system simulation. The simulation uses a channel model for two mutually interfering copper cables of length 1100 and 1000 feet. The 1% worst case interference model is used and VDSL standard guidelines are used for operating parameters. Upstream transmission is simulated, with 1174 subcarriers for both users. The starting states are generated by running ILC algorithm for varying number of iterations. Some other starting points are generated changing subcarrier powers by 3 dB and then allowing users to adapt independently after that. The actions are chosen randomly, independently for each user. Some simulation runs are done by fixing the action and always executing the same action no matter what state is reached. The procedure exits when the acceptable state is reached.

The *adaptation algorithm* takes as input: acceptable margin tolerance, maximum number of subcarriers allowed to be updated, the noise and channel couplings, the current bit and power allocation for the user. It returns the updated bit and power allocation for the user, fixing upto the maximum number of tones as needed without violating spectral mask, maximum bit constraint and total power constraints. The algorithm only swaps bits, it does not add or delete bits, thus maintaining the data rate. The algorithm however updates the power and allowed powers are not discretized.

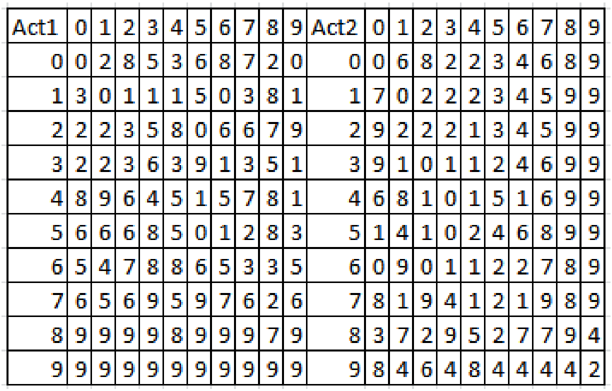
Both users do the 'action', i.e. run the adaptation algorithm, independently. Both users also have their states indepently. However the model takes into account a state pair and an action pair and combines it into one overall system state and action. Users act independently without knowledge of or coordination with the other user, but the end system state is dependent on both users' actions.

## V. RESULTS

Figure 1 gives the policy learnt by value iteration run on the model. The state for user 1 is mentioned in the row header and the state for user 2 is mentioned in the column header. The left side table gives the best learned action for user 1 for each possible state of the system and the right side table shows the best learned action for user 2. The results show that the best policy found with the given amount of learning does not discourage high rate of adaptation other than for the extreme case of action (9,9).

Figure 2 shows the result of running the system with the learned policy from 7 different starting points and counting the iterations. The red bar shows the average number of iterations needed to conveRege using the learned policy. This performance is compared against the performance of randomly chosen policies. The figure shows that the learned policy gives performance in the top 3rd percentile when compared with randomly chosen policies. This means, that although the learned policy is not the optimum policy, yet the algorithm has definitely learned something of value. Given more learning, the policy should get better.

The last figure shows a histogram of average iterations to convergence for the same 7 starting points. This time



| Act1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Act2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 8 | 5 | 3 | 6 | 8 | 7 | 2 | 0 | 0 | 0 | 6 | 8 | 2 | 2 | 3 | 4 | 6 | 8 | 9 |
| 1 | 3 | 0 | 1 | 1 | 1 | 5 | 0 | 3 | 8 | 1 | 1 | 7 | 0 | 2 | 2 | 2 | 3 | 4 | 5 | 9 | 9 |
| 2 | 2 | 2 | 3 | 5 | 8 | 0 | 6 | 6 | 7 | 9 | 2 | 9 | 2 | 2 | 2 | 1 | 3 | 4 | 5 | 9 | 9 |
| 3 | 2 | 2 | 3 | 6 | 3 | 9 | 1 | 3 | 5 | 1 | 3 | 9 | 1 | 0 | 1 | 1 | 2 | 4 | 6 | 9 | 9 |
| 4 | 8 | 9 | 6 | 4 | 5 | 1 | 5 | 7 | 8 | 1 | 4 | 6 | 8 | 1 | 0 | 1 | 5 | 1 | 6 | 9 | 9 |
| 5 | 6 | 6 | 6 | 8 | 5 | 0 | 1 | 2 | 8 | 3 | 5 | 1 | 4 | 1 | 0 | 2 | 4 | 6 | 8 | 9 | 9 |
| 6 | 5 | 4 | 7 | 8 | 8 | 6 | 5 | 3 | 3 | 5 | 6 | 0 | 9 | 0 | 1 | 1 | 2 | 2 | 7 | 8 | 9 |
| 7 | 6 | 5 | 6 | 9 | 5 | 9 | 7 | 6 | 2 | 6 | 7 | 8 | 1 | 9 | 4 | 1 | 2 | 1 | 9 | 8 | 9 |
| 8 | 9 | 9 | 9 | 9 | 8 | 9 | 9 | 9 | 7 | 9 | 8 | 3 | 7 | 2 | 9 | 5 | 2 | 7 | 7 | 9 | 4 |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 8 | 4 | 6 | 4 | 8 | 4 | 4 | 4 | 4 | 2 |

Fig. 1. Best Actions for State1/State2 for Users 1 and 2

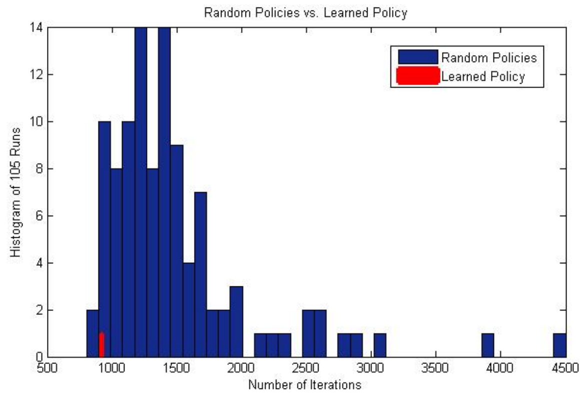


Fig. 2. Average Performance for Learned vs. Random Policies

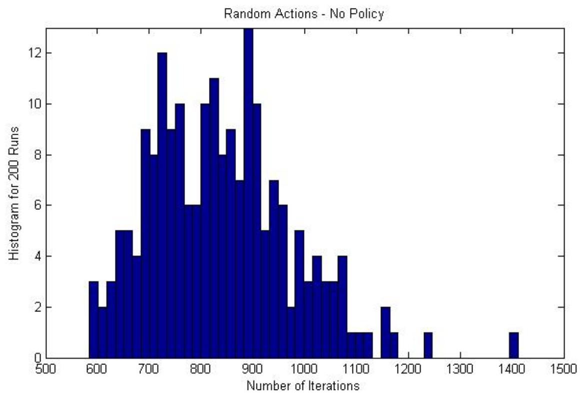


Fig. 3. Results Showing Possible Improvement

however, no policy is fixed and random actions are done in each iteration, each user independently picking its own policy randomly in each state. The low end of the histogram shows that a lot of iterations with randomly chosen actions perform better than any of the random policies or the learned policy. This means that assigning particular actions and states as done for this project was too restrictive. Given more flexibility, or more degrees of freedom in this case, given the same state and choosing different actions (random means different each time with a high probability), gives better performance than strict state-action pairing of the previous figure. The conlusion is that the state-action representation used was not granular enough. Nonetheless the results of figure 2 still show that it is possible for DSL systems to learn good policies for fast convergence.

## VI. Discussion

The project evolved from the following questions: Does allowing DSL modems to adapt their subcarrier bit and PSD allocations during showtime, to ensure good operating conditions, cause system instability instead? Should the rate of adaptation be controlled to avoild instability? The results of this project show that the learned policy does not necessarily try to limit the rate of adaptation, other than for the extreme case of both users adapting all of their tones. The policy obtained using value iteration also shows that modems can be allowed to update their spectra and good control policies can be found. This is a good result, however, the results found so far do not make the system converge fast enough.

A different control technique can be applied to ensure fast convergence. While the current algorithm for bit and spectrum update is the same for all users, an algorithm that marks users as preferred or not-preferred and uses different update procedures for each type, can ensure faster convergence and better operating points by prioritizing different subcarriers for different types of users.

## VII. Conclusion

The project uses an example of a 2-user DSL interference channel to show that de-centralized adaptation of bit and spectrum allocation can converge to a stable, acceptable state. It shows that an adaptation policy based on rate of adaptation does affect the speed of convergence and good policies can be learned using reinforcement learning. Using a more realistic state representation, a bigger and more flexible actions space, and a more thorough learning procedure, much better performance can be obtained. Even the much simplified state and actions space used gives a policy with promising results. The approach needs to be extended for more complex systems with realistic number of users. The project uses centralized learning for proof of concept. In addition, more advanced update algorithms can be used to speed up the process of convegence by making users be ‘polite’ to each other.

## References

[1] J. M. Cioffi, EE379C course reader, Stanford University, 2008. [Online]. Available: http://www.stanford.edu/class/ee379c

[2] *Very high speed digital subscriber line transceivers (VDSL2)*, ITU-T Recommendation G.993.2, Dec. 2011.

[3] W. Yu, W. Rhee, S. Boyd, and J. Cioffi, *Iterative Water-filling for Gaussian Vector Multiple Access Channels*, IEEE Trans. Inform. Theory, vol. 50, no. 1, pp. 145151, Jan. 2004.

[4] R. Ferrari, R. Lopes and J. M. T. Romano, *On the convergence of iterative discrete bitloading for autonomous spectrum management in DSL systems*, Simposi Brasileiro de Telecomunicaces - SBrT 2009.