A Clustering Approach to Autonomous Spectrum Balancing Using Multiple Reference Lines for DSL

Sean Huberman, Christopher Leung, and Tho Le-Ngoc
Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada
Email: sean.huberman@mail.mcgill.ca ; christopher.leung@mail.mcgill.ca ; tho.le-ngoc@mcgill.ca

Abstract—Crosstalk is the limiting factor of performance in Digital Subscriber Line systems. The use of Dynamic Spectrum Management (DSM) can reduce the effects of crosstalk. For DSM algorithms, there is an important trade-off between the complexity of an algorithm and its performance. This paper introduces a method of applying cluster theory to Autonomous Spectrum Balancing using Multiple Reference Lines in order to gain a favorable trade-off more consistently. It is shown that the achievable performance is very close to that of existing state-of-the-art DSM algorithms, while requiring significantly fewer time-consuming interference measurements.

Index Terms—Digital Subscriber Line (DSL), Dynamic Spectrum Management (DSM), resource allocation, power allocation.

I. INTRODUCTION

Dynamic Spectrum Management (DSM) has been shown to provide significant performance improvements over Static Spectrum Management (SSM) techniques. One of the first DSM algorithms introduced was Iterative Water-Filling (IWF) [1]. In IWF, each user selfishly maximizes their own data-rate until a non-cooperative Nash equilibrium point is reached.

IWF gives significant data-rate improvements over SSM techniques; however, in many situations, IWF leads to sub-optimal performance. The sub-optimality of IWF is caused by inefficient use of the frequency spectrum.

Two semi-centralized algorithms DSB [2] and SCALE [3] make use of message passing in order to solve for a locally-optimal power allocation. By doing so, they can achieve strong performance; however, they require some sort of centralized control for the message passing.

Two Autonomous Spectrum Balancing using Multiple Reference Lines (ASB-MRL) methods were proposed in, which build upon the original ASB algorithm [4]. The ASB-MRL approach constructs a virtual network, representative of the actual network, and allows each user to self-optimize using locally available information.

It was shown in [5] that the ASB-MRL approach could achieve strong performance while still operating at a low complexity and requiring few time-consuming interference measurements to converge. As such, [5] showed the potential benefits of using the ASB-MRL approach, but did not develop a definite method for selecting the virtual network of reference lines and their parameters.

This paper introduces an IWF-like approach that uses a constant Lagrange multiplier offset and can be blindly applied to any network. The offset acts as a per-tone quota for each user. The fact that it is constant during the optimization ensures that the algorithm runs at a near-IWF complexity and avoids the need for per-iteration message passing. The offsets for each user and frequency tone are calculated using the MRL virtual network.

The rest of this paper is organized as follows: Section II introduces the system model, Section III presents a method of applying ASB using MRL, Section IV presents the simulation results, and Section V provides some concluding remarks.

II. SYSTEM MODEL

Consider a DSL network with a set of users (modems) \( N = \{1, \ldots, N\} \) and a set of tones (frequency carriers) \( K = \{1, \ldots, K\} \). Using synchronous Discrete Multi-Tone (DMT) modulation, transmissions can be modeled independently on each tone \( k \) as \( y_k = H_k x_k + z_k \), where \( x_k = \{x^n_k, n \in N\} \) contains the transmitted signals, \( y_k = \{y^n_k, n \in N\} \) contains the received signals, and \( z^n_k \) is the additive white gaussian noise. \( h^n_{k,m} = [H]_{n,m} \) is the channel gain, \( s^n_k = \mathcal{E}\{|x^n_k|^2\}/\Delta_f \) is the transmit Power Spectral Density (PSD), for user \( n \) on frequency tone \( k \), respectively. \( \mathcal{E}\{\cdot\} \) denotes expected value, and \( \Delta_f \) denotes the frequency tone spacing. The vector containing the PSD of user \( n \) on all frequency tones is defined as \( s^n_k = (s^n_{1,k}, k \in K) \).

When the number of users is large enough, the interference is well approximated by a Gaussian distributed random variable, and hence the achievable bit rate of user \( n \) on frequency tone \( k \) is defined as:

\[
b^n_k \approx \log_2 \left( 1 + \frac{1}{\Gamma} \frac{|h^n_{k,n}|^2 s^n_k}{|s^n_k|^2 + \sigma^n_k} \right),
\]

where \( \Gamma \) is the signal to noise ratio gap which is a function of the desired bit error rate, coding gain, and noise margin [6], and \( \sigma^n_k \approx \mathcal{E}\{|x^n_k|^2\}/\Delta_f \) is the noise power density of user \( n \) on frequency tone \( k \). The achievable data rate for user \( n \) is therefore \( R^n = f_s \sum_k b^n_k \), where \( f_s \) is the DMT symbol rate.

This paper will focus on the following Rate Adaptive (RA) optimization problem:

\[
\max_{w^n, n \in N} \sum_{n \in N} w^n R^n
\]

subject to: \( \Delta_f \sum_{k \in K} s^n_k \leq P^n, \quad \forall \ n \quad (1) \)

\[
0 \leq s^n_k \leq s^n_{k,\text{mask}}, \quad \forall \ n, k,
\]

where \( \sum_{n \in N} w^n \leq 1 \) and \( w^n \) is the power allocated to user \( n \), \( P^n \) is the total transmitted power of user \( n \), \( \Delta_f \) is the frequency tone spacing, \( \sigma^n_k \) is the noise power density of user \( n \) on frequency tone \( k \), and \( s^n_{k,\text{mask}} \) is the maximum power allowed for each tone of user \( n \) including the mask power.
where $w^n_k$ is a weighting factor which represents the importance of user $n$. The first constraint limits user $n$’s total transmit power to $P^n_n$. The box constraint prevents negative power and adds the PSD mask $s^{n,mask}_k$.

### III. Methods of Applying MRL

Two algorithms using MRL were derived in [5], ASB-DSB and ASB-SCALE. These algorithms transformed DSB and SCALE from semi-centralized algorithms into fully-distributed ones that each user can run using locally-available information. Such information includes the measured interference and the reference lines’ channel and PSD. The Constant Offset ASB-MRL algorithm discussed in this paper builds on the ASB-DSB algorithm derived in [5].

#### A. Constant Offset ASB-MRL

For both the ASB-DSB and ASB-SCALE PSD update formulas derived in [5], the Lagrange multiplier offset needs to be re-computed for every iteration. The constant offset ASB-MRL algorithm assumes that the virtual network of reference lines is representative of the overall network and therefore the Lagrange multiplier offset can be approximately pre-computed before the optimization is run. As such, the Lagrange multiplier offset is a constant, and therefore the computational complexity of the algorithm can be reduced.

The constant offset ASB-MRL PSD update formula for user $n$ on frequency tone $k$ can be written as:

$$s^n_k = \left( \frac{w^n_k}{\lambda^n_k + \Delta \lambda^n_k} - \frac{\Gamma \int \text{int}^n_k}{|h^n_{k,n}|^2} \right) s^{n,mask}_k,$$

where $\text{int}^n_k = \sigma^n_k + \sum_{m \neq n} |h^n_{k,m}|^2 s^m_k$ is the interference seen by user $n$ and $\Delta \lambda^n_k$ is the Lagrange multiplier offset which must be estimated prior to the optimization. The constant offset ASB-MRL method estimates the Lagrange multiplier offset using MRL. In particular, the virtual network of reference lines is constructed algorithmically. This allows for the constant offset ASB-MRL to be applied to an arbitrary network without a priori knowledge of the network. The constant offset ASB-MRL algorithm can be broken up into the following steps.

1) **Setup Virtual Reference Line Network**: The first step is to assemble the virtual reference line network, which is a collection of reference lines aiming to represent the network. In order to determine how many reference lines should be selected, the key concept employed is clustering. For a particular network, it might be easier to group similar users based on observing the network topology; however, the approach presented in this paper has the practical advantage that it can be blindly applied to any arbitrary network topology. More specifically, it uses a payoff function to represent the relative strengths of each user in the network.

One-dimensional hierarchical clustering is applied to the results of the payoff function to group “similar users”. The clustering algorithm used in this paper is Matlab’s “clustdata” agglomerative hierarchical clustering function. This function starts by computing the pair-wise Euclidean distances between all elements. A hierarchical cluster tree is then built using the computed distance. Finally, the function performs clustering by using a cut-off point of 1 on the cluster tree.

Once the clusters are computed, each cluster is replaced by the reference line. The mean and median of each cluster are both reasonable choices. Typically, the mean and the median are similar; however, using medians rather than means makes the algorithm less sensitive to cluster outliers. The performance of the mean and median are shown in Section IV.

The payoff function produces a numerical value to represent how “weak” the user is. Typically, weaker users can be characterized as having larger received crosstalk gains relative to their direct channel gains. Similarly, stronger users can be characterized as having smaller crosstalk gains relative to their direct channel gains. As such, the payoff function was desired to be proportional to the crosstalk a user sees and inversely proportional to the users direct channel gain. This led to the ratio, $\sum_{m \neq n} |h^n_{k,m}|^2/|h^n_{k,n}|^2$. The payoff function was selected as:

$$XT_{\text{sum}}(n) \triangleq \sum_k \log \left( 1 + \frac{\sum_{m \neq n} |h^n_{k,m}|^2}{|h^n_{k,n}|^2/\Gamma} \right) = \sum_k \log \left( \frac{\sum_m |h^n_{k,m}|^2}{|h^n_{k,n}|^2} \right),$$

where $|h^n_{k,m}|^2 = |h^n_{k,n}|^2$ for $m \neq n$ is the crosstalk channel gain and $|h^n_{k,n}|^2 = |h^n_{k,n}|^2/\Gamma$ for all $n$ is the normalized direct channel gain, on tone $k$. Hence, $\sum_m |h^n_{k,m}|^2$ represents the received signal with crosstalk for user $n$ on tone $k$ when one unit of power is transmitted by each user and the background noise is negligible. Furthermore, $\sum_m |h^n_{k,m}|^2/|h^n_{k,n}|^2$ represents the ratio of the total received signal to the “useful” part of the received signal for user $n$ on tone $k$ when each user transmits identical power and the background noise is negligible. In particular, $\sum_m |h^n_{k,m}|^2/|h^n_{k,n}|^2$ represents the inverse Signal-to-Interference-plus-Noise Ratio (SINR$^{-1}$) when each user transmits identical power and the background noise is negligible. Therefore, the smaller the SINR$^{-1}$, the stronger the user, and the larger the SINR$^{-1}$, the weaker the user.

The log function is applied to the ratio for each frequency tone since it is very effective at differentiating between slight changes for small values (between strong users) and is relatively insensitive to slight changes for large values (between weak users). As such, it can easily differentiate between stronger users and penalize them accordingly. As well, it does not overly penalize weaker users that may be stronger than other weak users. Finally, $XT_{\text{sum}}(n)$ is calculated by summing over all the frequency tones. As such, each frequency tone (weighted equally) is factored into the payoff function for each user. A summary of the algorithm is shown in Algorithm 1.

2) **Select Parameters for Each Virtual Reference Line**: Once the reference line network is selected, based on the relative start and end points of each line, the ANSI model [7] can be used to generate the channel transfer functions for each reference line. The only remaining parameters that need to be computed are the PSDs of each reference line. To generate the PSD, [5] used Water-Filling and scaled down the resulting PSD; however, it used heuristics in order to determine the
The constant Lagrange multiplier offset algorithm has the same computation complexity as IWF. Once the clustering process is evaluated for each reference line, the method for computing the offset is initialized, the algorithm has the same computational complexity per iteration as IWF.

The constant offset ASB-MRL algorithm can be implemented in a distributed manner with an initialization phase where the Central Office (CO) computes each user frequency-selective Lagrange multiplier offsets. After each user receives their respective offsets, the optimization can be done in a fully distributed manner with a near-IWF complexity. The offsets provide global information. This typically allows for the constant offset ASB-MRL algorithm to achieve more favorable performance, while maintaining an IWF-like complexity. Due to the fact that the offsets are constant, no message passing during the optimization process is needed. This is a significant practical advantage over DSB and SCALE which both require message passing and recomputing the offsets at every iteration.

5) Convergence of the Constant Offset ASB-MRL:

**Theorem 1.** If \( \max_{n,k,m \neq n} \| h_{k,m}^n \| < \frac{1}{\sqrt{m-1}} \), the constant offset ASB-MRL algorithm using fixed weights, will converge to a unique fixed point using parallel updates.

**Proof.** The convergence proof for the constant offset ASB-MRL algorithm generalizes that of the ASB-S2 algorithm derived in [8]. The details of the proof are omitted but interested readers are referred to [9]. The convergence proof for the parallel update case uses the concept of a contraction mapping combined with Proposition 1.1 of [10], which states that the sequence of iterates formed by a contraction mapping geometrically converges to a unique fixed point.

IV. SIMULATIONS

Typically, simulations are run for a particular network configuration. Based on the particular network configuration, the performance of various DSM algorithms are compared. The danger with this approach is that it can lead to inaccurate conclusions due to the fact that different scenarios favor some algorithms more than others. As such, adopting a Monte-Carlo simulation style, where various network configurations are
tested but the specific network parameters are varied, can lead to more accurate overall comparisons.

This section will discuss three different test cases representing practical deployments. Each test case consists of a 25-user Monte-Carlo simulation where the distribution of users line lengths are assumed to be uniform (i.e., the number of users that have longer line lengths is similar to the number of users that have short line lengths).

The three 25-user Monte-Carlo test cases were simulated consisting of 1000 upstream and 1000 downstream network realizations. The scenarios are described in Table I. The notation “X users from A – [B, C] ft” means that the X users are offset by A feet and their line length is distributed (either uniformly or exponentially) between [B – A, C – A].

### Table I
**SUMMARY OF TEST CASES**

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>All CO</th>
<th>CO-RT</th>
<th>CO-RT-RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>All CO</td>
<td>25 users from 0 – [1500, 3000] ft</td>
<td>13 users from 0 – [1000, 2000] ft</td>
<td>9 users from 0 – [1000, 2000] ft</td>
</tr>
<tr>
<td>CO-RT</td>
<td>12 users from 500 – [1500, 2500] ft</td>
<td>8 users from 500 – [1500, 2500] ft</td>
<td>8 users from 1000 – [1500, 2500] ft</td>
</tr>
<tr>
<td>CO-RT-RT</td>
<td>9 users from 0 – [1000, 2000] ft</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All test cases assume that 26-gauge (0.4 mm) lines are used. The target symbol error probability is set at $10^{-7}$. The coding gain and noise margin are set to 3 dB and 6 dB, respectively. The frequency tone spacing is $\Delta f = 4.3125$ kHz and the DMT symbol rate is $f_s = 4$ kHz. PSD masks were applied using VDSL Profile 17a band plan [7]. A maximum transmit power of 11.5 dBm is applied to each modem.

For each network realization, DSB, SCALE, IWF, ASB-MRL-mean and ASB-MRL-median were computed. The performance of the algorithms was compared based on the percent of the maximum sum rate each algorithm achieved, on a per-realization basis. As such, the average percent of maximum represents the expected performance relative to the other DSM algorithms tested for each particular network configuration.

The maximum and minimum percentages and the standard deviations were also computed. These quantities represent the possible variable between network realizations. In particular, the variation in these quantities provides information about the stability and robustness of the algorithms and demonstrates the need for using Monte Carlo simulations.

The results also provide the number of iterations. The time per iteration of most distributed DSM algorithms are relatively fast and as such, the limiting factor in terms of runtime becomes the number of iterations. As well, each user must take new interference measurements after each iterative. Therefore, it is desired to reduce the number of iterations. The number of iterations required for each algorithm to achieve 98% of their final converged rate is also given. This value is relevant for practical systems where the goal is to minimize the number of iterations while still operating close to the algorithms’ actual performance (e.g., 98% of final converged sum rate).

### A. All CO Test Case

The results for the All CO test case are shown in Table II. DSB and SCALE achieved the maximum performance in both the upstream and downstream settings. For the upstream test case, it can be seen that using ASB-MRL can reduce the average number of iterations from 142.0 for DSB and 190.7 for SCALE to 19.7 for ASB-MRL, while still achieving 97.8% of the maximum performance on average. As well, the number of iterations to reach 98% of the final converged value drops from approximately 11 (for DSB/SCALE) to 2. Based on the minimum percent of the maximum performance, it is clear that the performance of ASB-MRL was always greater than 92% of the maximum value, while IWF was as small as approximately 70% of the maximum value.

For the All CO downstream test case, all the algorithms performed strongly; however, the number of iterations required for ASB-MRL (24.9 or 28.7, on average) were significantly less than the required number of iterations for DSB and SCALE (133.7 and 288.7 on average, respectively). The average number of iterations to achieve 98% of the final converged value for DSB was only 2.8; however, SCALE required 19.5 iterations to achieve 98% of its final converged performance.

### B. CO-RT Test Case

Simulations for the CO-RT test case are shown in Table III. DSB and SCALE both achieved an average of 100% of the maximum for the upstream setting, while in the downstream scenario SCALE achieved 100% of the maximum and DSB achieved 99.6% of the maximum.

For the CO-RT upstream test case, DSB and SCALE required 246.6 and 274.2 iterations on average to converge. ASB-MRL required only 13.9 while still achieving an average performance of 99.1% of the maximum. In particular, ASB-MRL required only 2 iterations to achieve 98% of the converged result, whereas DSB and SCALE required 17.4 and 19.5 iterations on average, respectively. When comparing the minimum percent of the maximum, it can be seen that ASB-
MRL dropped to as low as 84.3%; however, this is still a significant gain over IWF’s 65.5%.

For the CO-RT downstream test case, DSB and SCALE required 382.8 and 310.6 iterations on average to converge. ASB-MRL required only 199 iterations to converge, while still maintaining 99.6% of maximum performance, on average. Again, the number of iterations to achieve 98% of the final converged rate was only 2 for ASB-MRL, but 11.6 and 27.9 for DSB and SCALE on average, respectively.

### C. CO-RT-RT Test Case

The simulation results for the CO-RT-RT test case are shown in Table IV. For the upstream test case, ASB-MRL achieved 99.9% of the maximum while requiring only 10.6 iterations, on average. DSB and SCALE achieved 100% of the maximum but required 194.9 and 245.3 iterations on average, respectively. Similarly, ASB-MRL required only 2 iterations to achieve 98% of its final converged performance, as compared to 15.6 and 38.1 for DSB and SCALE on average, respectively.

For the CO-RT-RT downstream test case, DSB, SCALE and ASB-MRL all achieved over 99% of maximum while ASB-MRL required only 32 iterations for convergence on average, as compared to 394.4 and 241.9 for DSB and SCALE, respectively. Again, ASB-MRL required only 2 iterations to achieve 98% of the final converged rate, whereas DSB and SCALE required 14.5 and 19 on average, respectively.

### V. Conclusion

There is an important trade-off between the achievable data-rate (performance) and the complexity. The constant offset ASB-MRL algorithm was presented. The construction and development of a crosswalk sum payoff function was discussed. The ASB-MRL algorithm uses clustering on the crosswalk sum payoff function to algorithmically construct a virtual network of reference lines. The crosswalk sum payoff function was also used to more accurately approximate the PSD of each reference line. A uniqueness and existence theorem for the ASB-MRL algorithm was also provided.

A Monte-Carlo simulation technique was introduced. Based on the simulation results, it can be seen that there can be a large variation in the relative performance of some DSM algorithms between similar, yet different, network realizations. These results further support the need for the more rigorous Monte-Carlo simulation technique.

The simulation results showed that the constant offset ASB-MRL algorithm could achieve performance near DSB and SCALE while requiring significantly fewer iterations. Reducing the number of iterations drastically affects the runtime of the algorithm since at each iteration new interference measurements need to be taken. Therefore, the constant offset ASB-MRL algorithm provides a significant improvement in the trade-off between performance and complexity by achieving near-DSB and SCALE performance at a near-IWF complexity.

### REFERENCES


