DSM Algorithms in Multi-User xDSL

Sean Huberman, Christopher Leung, Tho Le-Ngoc

August 26, 2009

Abstract

Dynamic Spectrum Management (DSM) is an effective method for reducing the effect of crosstalk in Digital Subscriber Line (DSL) systems. This technical report discusses seven DSM algorithms: Optimal Spectrum Balancing (OSB), Iterative Spectrum Balancing (ISB), Autonomous Spectrum Balancing (ASB), Iterative Water-Filling (IWF), Selective Iterative Water-filling (SIW), Successive Convex Approximation for Low complExity (SCALE), and the simple Difference of Convex function Algorithm (DCA). Their performance (achievable data rate) and computational complexity are discussed.

Keywords: Digital Subscriber Line (DSL), Dynamic Spectrum Management (DSM), resource allocation, non-convex optimization.

1 Introduction

Digital Subscriber Line (DSL) continues to be the most popular broadband access technology [1]. The most significant factor in the performance of DSL systems is the interference between cables, known as crosstalk. Crosstalk has the potential to severely limit the performance of the system if it is not dealt with. The DSL channel is highly frequency-selective and, in general, at high frequency the crosstalk is very significant. In order to make use of the higher frequency bands, effective spectrum management techniques must be employed. The most basic form of spectrum management is Static Spectrum Management (SSM) [2]. SSM implements identical spectral masks based on a worst-case scenario assumption for all users. Clearly, this leads to inefficient spectrum management whenever the scenario is not the worst-case and leads to highly sub-optimal performance.

The poor performance of SSM systems led to the introduction of Dynamic Spectrum Management (DSM) [3]. DSM is a wide field which looks to adaptively apply different spectral masks for each user with the intent of maximizing the throughput of the system. DSM allows for a far more efficient use of the spectrum than SSM does. As a result, many different DSM algorithms have been proposed.

The main criteria in comparing different DSM algorithms are their performance and their complexity. The performance relates how well an approach succeeds at maximizing the achievable data rate as compared to the theoretical optimum. The complexity of the model is related to the amount of time required to derive the power allocation as the number of users and frequency tones increase.

There are two main types of DSM algorithms: centralized and distributed. Centralized systems require a central hub with full knowledge of the network. In general, this system allows for better performance at a cost of increasing the complexity and computational time. On the other hand, distributed systems allow for every user to self-optimize. Distributed systems that can operate fully autonomously without the need of explicit message passing are called autonomous systems. In general, distributed systems reduces the complexity and computational time but often sacrifices some optimality in terms of performance.

One of the first distributed DSM algorithms was Iterative Water Filling (IWF) [4]. In IWF, each user selfishly maximizes their own data-rate. While IWF gives significant data-rate improvements over SSM techniques, in many situations, it leads to sub-optimal performance.

The sub-optimality of IWF is caused by inefficient use of the frequency spectrum. In an attempt to increase the efficiency of the frequency spectrum, another algorithm known as Selective Iterative Water-filling (SIW) [5] was introduced. SIW selectively applies IWF to users in the under-utilized sections of the frequency spectrum until all users use up their total power. While SIW shows significant performance gains over IWF, the performance is still sub-optimal.

A centralized algorithm called Optimal Spectrum Balancing (OSB) [6], which maximizes the weighted rate sum across the users, was derived to solve for the globally optimal power allocation. OSB uses dual decomposition to solve for the optimal transmit powers for each user separately on each frequency tone by exhaustive search. While OSB is not computationally tractable for many users, it serves as an upper bound on the performance of other DSM algorithms for cases with few users.

Since OSB is intractable for many users, a centralized algorithm called Iterative Spectrum Balancing (ISB) [7] [8] was introduced. ISB formulates the optimization problem similar to OSB, using the weighted rate sum and dual decomposition, but solves for the power allocations in an iterative fashion using one-dimensional instead of N-dimensional exhaustive searches (where N is the number of users). This allows for ISB to be computationally tractable for many users.

OSB and ISB are centralized systems, which are generally harder to implement in practice. It is for this reason that many new algorithms were introduced. One such algorithm is Autonomous Spectrum Balancing (ASB) [9]. ASB uses the same problem formulation as OSB and ISB but operates in a distributed fashion without the need for any explicit message passing. ASB uses the concept of a virtual reference line, which represents the typical victim in the network. Each user self-optimizes to protect the reference line and hence attempts to better the overall network. Generally ASB cannot find a globally optimal solution; however, its performance has been shown to be near-optimal in many situations while maintaining a relatively low complexity.

Another algorithm called Successive Convex Approximation for Low-complExity (SCALE) [10] [11] can be viewed as a hybrid centralized and distributed system since it can be implemented in a distributed fashion with explicit message passing. SCALE is an algorithm that applies a series of concave lower-bounds to the maximization problem. This enables SCALE to make use of the well researched area of convex optimization to maximize the concave lower-bound. Each successive iteration tightens the lower-bound towards the locally optimal solution.

By representing the objective function of the non-convex optimization problem in OSB explicitly as the Difference of two Convex functions (DC) [12], (i.e., $f = g - h$), a modified prismatic branch-and-bound algorithm was proposed in [13]. The constraint set is convex since it consists of a system of linear equations and one nonlinear constraint. The Prismatic Branch and Bound (PBnB) algorithm [14] operates by successively approximating the nonlinear constraint by a piecewise linear function, and finds a globally optimal solution by solving a sequence of Linear Programming (LP) sub-problems. Although its computational complexity is still high, the algorithm proposed in [13] can substantially reduce the complexity of OSB in finding a globally optimal solution, especially for large number of users. It can provide an upper-bound on the performance in many situations where OSB is computationally intractable, and hence, can serve as a comparison measure for other more practical DSM algorithms.

Another algorithm discussed in this technical report makes use of the simple DC Algorithm (DCA) [15]. DCA is a centralized algorithm that begins by re-writing the non-convex objective function in terms of the difference of two convex functions (i.e., $f = g - h$); however, DCA iteratively creates an affine minorization (multivariate first-order approximation) of h, denoted by $\tilde{h}$, which is used to make the objective function, $\tilde{f} = g - \tilde{h}$, convex. Each successive iteration more closely approximates the locally optimal solution. For any function f, many DC decompositions exist (e.g., $g - h = (g + \phi) - (h + \phi)$). The choice of decomposition has a crucial impact on the convergence speed as well as the performance. There are still a lot of heuristics regarding the DCA implementation which have yet to be explored in great detail.

The rest of this technical report is organized as follows: Section 2 presents a brief overview of DSM problems in multi-user xDSL. Section 3 discusses the seven algorithms: OSB, ISB, ASB, IWF, SIW, SCALE, and DCA. Section 4 presents an illustrative example and simulation results to evaluate and compare the performance of the seven algorithms, while Section 5 discusses the complexity behaviour of the algorithms under consideration. Section 6 provides some concluding remarks.

2 DSM in Multi-User DSL Network

2.1 xDSL Environment

xDSL is a family of technologies which transmit digital data over twisted-pair copper telephone wires [3] [16]. xDSL operates on the same twisted-pair copper wire as Plain Old Telephone Service (POTS) since it uses a higher frequency band, while POTS uses a lower frequency band (less than 3.4 kHz). The frequency band used for DSL depends on the specific technology. For example, in Asymmetric Digital Subscriber Line (ADSL) the maximum frequency used is 1.1 MHz, in ADSL2+ the maximum frequency used is 12 MHz, and in Very high bitrate DSL (VDSL) the maximum frequency used is 30 MHz. There are dedicated frequency bands for upstream and downstream transmission. For most DSL technologies, a larger frequency band is allocated for downstream transmission than for upstream transmission.

Twisted-pair copper wires are run underground in binder groupings. Typically, the maximum number of twisted-pair copper wires in a binder is 200. The number of frequency tones depends on the type of DSL technology used. For example, ADSL uses 256 frequency tones, whereas VDSL uses 4096 frequency tones.

2.2 System Model

Consider a DSL network with a set of users (modems) $\mathcal{N} = \{1, \ldots, N\}$ and a set of tones (frequency carriers) $\mathcal{K} = \{1, \ldots, K\}$. Using synchronous Discrete Multi-Tone (DMT) modulation, there is no Inter-Channel Interference (ICI) and transmissions can be modeled independently on each tone k as follows:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{z}_k.$$

The vector $\mathbf{x}_k \triangleq \{x_k^n, n \in \mathcal{N}\}$ contains the transmitted signals for all users on frequency tone k, where x_k^n is the transmitted signal by user n on frequency tone k. Similarly, $\mathbf{y}_k \triangleq \{y_k^n, n \in \mathcal{N}\}$ and $\mathbf{z}_k \triangleq \{z_k^n, n \in \mathcal{N}\}$ where y_k^n is the received signal for user n on frequency tone k. Likewise, z_k^n is the additive noise for user n on frequency tone k which contains thermal noise, alien crosstalk and radio frequency interference. $\mathbf{H}_k$ is an $N \times N$ matrix such that $[\mathbf{H}_k]_{n,m}$ is the channel gain from transmitter m to receiver n on frequency tone k, and is defined as $h_k^{n,m}$. The transmit Power Spectral Density (PSD) of user n on frequency tone k is defined as $s_k^n \triangleq \mathcal{E}\{|x_k^n|^2\}/\Delta_f$, where $\mathcal{E}\{\cdot\}$ denotes expected value, and Δ_f denotes the frequency tone spacing. The vector containing the PSD of user n on all frequency tones is defined as $\mathbf{s}^n \triangleq \{s_k^n, k \in \mathcal{K}\}$.

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

$$b_k^n \triangleq \log_2 \left(1 + \frac{1}{\Gamma} \frac{|h_k^{n,n}|^2 s_k^n}{\sum_{m \neq n} |h_k^{n,m}|^2 s_k^m + \sigma_k^n} \right)$$

where Γ is the Signal to Noise Ratio (SNR) gap which is a function of the desired Bit Error Rate (BER), coding gain, and noise margin [16], and $\sigma_k^n \triangleq \mathcal{E}\{|z_k^n|^2\}/\Delta_f$ is the noise power density of user n on frequency tone k. The achievable data rate for user n is therefore

$$R^n = f_s \sum_k b_k^n.$$

2.3 Spectrum Management Problem

The spectrum management problem, based on the system model shown in Section 2.2, involves selecting the transmit powers for each of the N users on each of the K frequency tones.

The Static Spectrum Management (SSM) approach assumes a worst-case scenario where all the possible users in a cable binder are active [2] [3]. Based on this assumption, the interference plus noise term for each user does not change (i.e., it is static). Each user n must then adjust their transmit power, s_k^n, on each frequency tone k with the assumption that the interference plus noise seen is at its maximum.

There are many different types of SSM technologies. One example is called flat Power Back-Off (PBO). Flat PBO operates by ensuring that each user transmits the minimum possible flat PSD required to meet its Quality of Service (QoS) requirement. Another example is the reference noise method where each user sets its transmit PSD so that the crosstalk felt by the virtual modem is equal to the background noise seen by the virtual modem.

When all users are active and the situation is a worst-case model, SSM provides reasonable spectrum management; however, whenever the total number of active users is less than the total number of users in the cable binder, SSM becomes overly pessimistic and makes inefficient use of the spectrum. The worst-case model used by SSM is entirely determined based on the length and width of the cable used.

In practice, many bridge-taps, splices, and other cable imperfections exist, which make the SSM worst-case model even more inaccurate. As well, in practice, the number of active users changes constantly and therefore in order to make efficient use of the spectrum, it is necessary to dynamically allocate transmit PSDs in order to maximize the achievable bit-rate.

Dynamic Spectrum Management (DSM) dynamically allocates transmit powers for all users in response to changes in channel conditions [3]. There are several physical constraints imposed on the transmit powers for each user. One such constraint is the maximum power which each user is allowed to allocate over all of its frequency tones. The maximum power constraint for user n is denoted by P^n. Another constraint is the maximum power which each user is allowed to allocate on any particular frequency tone referred to as a spectral mask. The spectral mask for user n on frequency tone k is denoted by $s_k^{n,\text{mask}}$. Therefore the power constraints can be summarized as:

$$\sum_{k \in \mathcal{K}} s_k^n \leq P^n \qquad \text{and} \qquad 0 \leq s_k^n \leq s_k^{n,\text{mask}} \qquad \forall \quad n \in \mathcal{N}, \quad k \in \mathcal{K}. \tag{1}$$

There are many different possible spectrum management problems depending on the specific goal of the network. For example, maximizing the total achievable data rate (i.e., Rate Adaptive (RA)) or minimizing the total power allocated (i.e., Fixed Margin (FM)) while still ensuring some QoS requirements for each DSL customer. Regardless of the specific goal of the network, the physical constraints shown in (1) are always present.

One approach to DSM focuses on maximizing the achievable data rate of one particular user, given that the rest of the users satisfy some QoS requirement. The QoS requirement for user n is denoted by $R^{n,\text{target}}$. This approach to DSM can be summarized in the following RA optimization problem:

$$\begin{array}{ll} \max\limits_{\mathbf{s}^n,\ n \in \mathcal{N}} & R^1 \\ \text{subject to:} & R^n \geq R^{n,\text{target}}, \quad \forall \quad n > 1 \\ & \sum\limits_{k \in \mathcal{K}} s_k^n \leq P^n, \quad \forall \quad n \\ & 0 \leq s_k^n \leq s_k^{n,\text{mask}}, \quad \forall \quad n, k. \end{array} \tag{2}$$

It was shown in [6] that the optimal solution to the RA optimization problem (2) is equivalent to the optimal solution of the RA optimization problem (3) for some $\mathbf{w} \triangleq \{w^n, n \in \mathcal{N}\}$. w^n is a weighting factor which represents the importance of user n.

$$\begin{array}{ll} \max\limits_{\mathbf{s}^n,\ n \in \mathcal{N}} & \sum\limits_{n \in \mathcal{N}} w^n R^n \\ \text{subject to:} & \sum\limits_{k \in \mathcal{K}} s_k^n \leq P^n, \quad \forall \quad n \\ & 0 \leq s_k^n \leq s_k^{n,\text{mask}}, \quad \forall \quad n, k. \end{array} \tag{3}$$

In the most general form, one can incorporate both RA and FM into the spectrum management problem [11]. Let $\mathcal{RA}$ and $\mathcal{FM}$ denote the index sets of the RA and FM users, respectively. Assuming $\mathcal{RA} \neq \emptyset$ (i.e., there are some users which are trying to maximize their rate) the joint RA and FM spectrum management problem is shown below in (4). Note that the rate for all of the users in $\mathcal{RA}$ are being maximized while the rate for all of the users in $\mathcal{FM}$ are being fixed at their respective QoS requirement in order to minimize their power consumption.

$$\begin{array}{lll}
\max\limits_{\mathbf{s}^n,\ n\in\mathcal{N}} & \sum\limits_{n\in\mathcal{RA}} w^n R^n & \\
\text{subject to:} & R^n = R^{n,\text{target}}, \quad \forall \quad n\in\mathcal{FM} & (4)\\
& \sum\limits_{k\in\mathcal{K}} s_k^n \le P^n, \quad \forall \quad n\in\mathcal{N} & \\
& 0\le s_k^n \le s_k^{n,\text{mask}}, \quad \forall \quad k\in\mathcal{K},\ n\in\mathcal{N} &
\end{array}$$

If $\mathcal{RA} = \emptyset$ (i.e., no users are trying to maximize their rate) the spectrum management problem reduces to the FM case. This is shown below in (5). Note that $\mathcal{RA} = \emptyset \iff \mathcal{N} = \mathcal{FM}$ and the maximization problem changes to a minimization problem. Here the total power consumed by all users on all frequency tones is being minimized while ensuring each user still meets their respective QoS requirements.

$$\begin{array}{lll}
\min\limits_{\mathbf{s}^n,\ n\in\mathcal{N}} & \sum\limits_{n\in\mathcal{N}}\sum\limits_{k\in\mathcal{K}} s_k^n & \\
\text{subject to:} & R^n \ge R^{n,\text{target}}, \quad \forall \quad n\in\mathcal{N} & (5)\\
& \sum\limits_{k\in\mathcal{K}} s_k^n \le P^n, \quad \forall \quad n\in\mathcal{N} & \\
& 0\le s_k^n \le s_k^{n,\text{mask}}, \quad \forall \quad k\in\mathcal{K},\ n\in\mathcal{N} &
\end{array}$$

3 DSM Algorithms under consideration

The seven DSM algorithms: OSB, ISB, ASB, IWF, SIW, SCALE, and DCA are discussed. OSB is a centralized algorithm which computes the globally optimal solution by using N-dimensional exhaustive searches. ISB is a centralized algorithms which attempts to approximate OSB by iteratively performing 1-dimensional exhaustive searches.

ASB is an autonomous algorithm which uses the concept of a reference line to maximize the achievable data rate. IWF is an autonomous algorithm where every user selfishly maximizes their own transmit power, until the point is reached where no user can benefit from changing their transmit power. SIW iteratively performs IWF, which allows for weaker users to make more efficient use of the frequency spectrum.

SCALE is a hybrid centralized and distributed algorithm since it is implemented in a distributed fashion by making use of explicit message passing to gain centralized knowledge. SCALE operates by successively applying convex lower bounds to the objective function until a locally optimal point is reached.

DCA is a centralized algorithm which makes use of a Difference of Convex functions (DC) property in order to solve for a locally optimal points in an iterative fashion.

3.1 Optimal Spectrum Balancing (OSB)

As discussed in Section 5, solving for the optimal power allocation of all N users on each of the K frequency tones is $\mathcal{O}(e^{KN})$.

OSB begins by re-writing the optimization problem shown in (2) as the optimization problem shown in (3) [6] [7]. The OSB algorithm then uses the concept of dual decomposition to eliminate the exponential complexity in terms of the number of frequency tones, K, and converts it into a linear complexity in terms of the number of frequency tones, K. OSB converts the constrained optimization problem in (3) into an unconstrained optimization problem using duality theory. Therefore, the Primal Problem, (3), is replaced by the dual problem

$$\max_{s^1,\dots,s^N} \mathcal{L}(s^1,\dots,s^N) \tag{6}$$

$$\text{where } \mathcal{L}(s^1,\dots,s^N) \triangleq \sum_n w^n R^n - \sum_n \sum_k \lambda^n s_k^n.$$

The Lagrange multipliers $(w^1,\dots,w^N,\lambda^1,\dots,\lambda^N)$ are chosen such that the Karush–Kuhn–Tucker (KKT) conditions are satisfied:

$$\lambda^n \Big(P^n - \sum_k s_k^n \Big) \quad = 0, \quad \forall\, n \tag{7}$$

$$w^n \Big(R^n - \sum_k b_k^n \Big) \quad = 0, \quad \forall\, n. \tag{8}$$

Provided that Equations (7) and (8) hold, the dual problem (6) is equivalent to the primal optimization problem (3).

Since synchronous transmission is assumed, in Section 2.2, there is no ICI and therefore the Lagrangian can be decomposed over the frequency tones. This reduces the complexity of the algorithm from exponential in the number of frequency tones to linear in the number of frequency tones, since the exhaustive search can be performed independently on each frequency tone. Therefore, OSB reduces a KN-dimensional exhaustive search to K different N-dimensional exhaustive searches.

Therefore, for each frequency tone k, the following optimization problem must be solved:

$$\max_{s_k^1,\dots,s_k^N} \mathcal{L}_k(s_k^1,\dots,s_k^N) \tag{9}$$

$$\text{where } \mathcal{L}_k(s_k^1,\dots,s_k^N) \triangleq \sum_n w^n R^n - \sum_n \lambda^n s_k^n.$$

Since $\sum_k \mathcal{L}_k(s_k^1,\dots,s_k^N) = \mathcal{L}(s^1,\dots,s^N)$, solving (9) independently over each frequency tone is equivalent to solving (6). The process of writing out the Lagrangian dual problem and decomposing the Lagrangian across all the frequency tones is known as dual decomposition.

The full OSB algorithm is outlined in Algorithm 1. One important note is that the choice of ϵ can drastically effect the performance of the algorithm. If ϵ is too small, the algorithm will take an extremely long time to converge but if ϵ is too large the algorithm may not converge at all. Ideally one must tweak the value of ϵ heuristically in order to ensure proper convergence of the algorithm. This is often a highly non-trivial task.

Algorithm 1: OSB Algorithm

```
repeat
    foreach $k$ do
        Solve by $N$-Dimensional exhaustive search:
        $(s_k^1, \ldots, s_k^N) = \arg\max_{\{s_k^1,\ldots,s_k^N\}} \mathcal{L}_k(s_k^1, \ldots, s_k^N)$ ;
    end
    foreach $n$ do
        $w^n = \max\{w^n + \epsilon\ (R^{n,\text{target}} - \sum_k b_k^n),\ 0\}$ ;
        $\lambda^n = \max\{\lambda^n + \epsilon\ (\sum_k s_k^n - P^n),\ 0\}$ ;
    end
until $s_k^n$ converges $\forall\ k,\ n$ ;
```

Spectral masks can easily be incorporated into Algorithm 1 by setting the value of $\mathcal{L}_k$ to a very large negative number if $s_k^n > s_k^{n,\text{mask}}$ for each user n.

By reducing the complexity to linear in terms of the number of frequency tones, OSB becomes computationally tractable for large numbers of frequency tones; however, OSB is still computationally intractable for large numbers of users. OSB can therefore serve as an upper-bound with which to compare the performance of other DSM algorithms for few users.

3.2 Iterative Spectrum Balancing (ISB)

In Section 3.1, it is shown that OSB is computationally intractable for large N. This motivates the choice of an iterative algorithm which is tractable for large N. ISB operates in a similar fashion to OSB by re-writing the optimization problem shown in (2) as the optimization problem shown in (3) [7] [8]. ISB also uses the concept of dual decomposition, discussed in Section 3.1, to re-write the optimization problem as,

$$\max_{s_k^n} \mathcal{L}_k(s_k^1, \ldots, s_k^N) \tag{10}$$

$$\text{where } \mathcal{L}_k(s_k^1, \ldots, s_k^N) \triangleq \sum_n w^n R^n - \sum_n \lambda^n s_k^n.$$

Note that in ISB, the PSD of each user is searched for in an iterative fashion by updating one user's PSD at a time while keeping all other users' PSD values fixed. This can be seen in Equation (10) since the maximization is only over a single user's PSD, s_k^n. It is important to note that for ISB, the KKT conditions shown in Equations (7) and (8) must still be satisfied in order to ensure that the solution to the dual problem (10) for all k, is equivalent to the solution of the primal problem (3).

The full ISB algorithm is outlined in Algorithm 2. One important note is that the choice of ϵ can drastically effect the performance of the algorithm. If ϵ is too small, the algorithm will take an extremely long time to converge but if ϵ is too large the algorithm may not converge at all. Ideally one must tweak the value of ϵ heuristically in order to ensure proper convergence of the algorithm. This is often a highly non-trivial task.

Algorithm 2: ISB Algorithm

```
Let $\epsilon_R > 0$ and $\epsilon_P > 0$ be given ;
repeat
    foreach $n$ do
        repeat
            foreach $k$ do
                Fix $s_k^m \; \forall \; m \neq n$ ;
                Solve by 1-Dimensional exhaustive search:
                $s_k^n = \arg\max_{s_k^n} \mathcal{L}_k(s_k^1, \ldots, s_k^N)$ ;
            end
            $w^n = \max\{w^n + \epsilon_R \; (R^{n,\text{target}} - \sum_k b_k^n), \; 0\}$ ;
            $\lambda^n = \max\{\lambda^n + \epsilon_P \; (\sum_k s_k^n - P^n), \; 0\}$ ;
        until $w^1, \ldots, w^N, \lambda^1, \ldots, \lambda^N$ converge ;
    end
until $s_k^n$ converges $\forall \; k, \; n$ ;
```

Spectral masks can be easily incorporated into Algorithm 2 by setting the value of $\mathcal{L}_k$ to a very large negative number if $s_k^n > s_k^{n,\text{mask}}$ for each user n.

While ISB can not guarantee to find optimal power allocations, it has been shown to lead to optimal power allocations in many test cases, especially with few users. Unlike OSB, ISB has a complexity which is quadratic in the number of users and linear in the number of frequency tones therefore it is computationally tractable for many users. Even though it cannot guarantee to find the optimal power allocations, it can be used as a comparative measure since its performance is known to be close to optimal.

A similar algorithm called Generalized Iterative Spectrum Balancing (GISB) [17] was proposed. GISB operates similarly to ISB and OSB, where the difference lies in terms of the number of PSDs over which the exhaustive search is performed. For GISB, the exhaustive search is performed over L users where $1 \leq L \leq N$. Note that ISB and OSB are the special cases where $L = 1$ and $L = N$, respectively. GISB allows for a tradeoff between performance and computational complexity. The closer the value of L is to N, the better the performance will be at the expensive of computational complexity.

3.3 Autonomous Spectrum Balancing (ASB)

In Section 1, several distributed DSM algorithms were discussed. One main issue with distributed algorithms is that in some algorithms (e.g., IWF), each user selfishly self-optimizes which often leads to a Nash Equilibrium point (a point when no user can gain anything by changing its transmit power), which is highly sub-optimal. Autonomous algorithms are distributed algorithms which require no explicit message passing. The idea behind ASB is that each user tries to maximize the throughput of the system while ensuring that their performance is above some threshold value. This is a more selfless system and hence the overall performance would be expected to improve since each user is no longer acting selfishly.

The ASB algorithm [9] uses the concept of a virtual reference line which attempts to mimic a typical victim line in the interference channel. Each user then tries to minimize the damage done to the reference line while ensuring that its own target data-rate is met. The reference line is determined using statistics of the network which are known a priori and hence the ASB can be implemented autonomously. ASB makes use of the fact that DSL crosstalk channel gains are slowly time-varying and hence the reference line can represent a typical victim line.

One choice for the reference line is the longest line within the network, since the longest line tends to have the weakest direct transfer function and therefore is most sensitive to crosstalk. The only knowledge a modem needs is its own direct channel, background noise and the distance from the CO to the Remote terminal (RT, assuming it is RT distributed). All these values can be measured locally or programmed at the time the RT is installed. This allows for ASB to operate fully autonomously during run-time.

From user n's point of view, the reference line's rate is:

$$R^{n,\mathrm{ref}} \triangleq \sum_k \tilde{b}_k^n,$$

where

$$\tilde{b}_k^n \triangleq \log_2\left(1 + \frac{1}{\Gamma}\frac{|\tilde{h}_k^{\tilde{n},\tilde{n}}|^2 \tilde{s}_k}{|\tilde{h}_k^{\tilde{n},n}|^2 s_k^n + \tilde{\sigma}_k}\right).$$

The crosstalk channel for the reference line, denoted $|\tilde{h}_k^{\tilde{n},\tilde{n}}|^2$ and $|\tilde{h}_k^{\tilde{n},n}|^2$, are modeled using empirical models (ANSI models) that were developed in the standards [18], [19], and [20]. When using this model, the only parameters required to determine the crosstalk channel values are the length of the coupling line, offsets (network topology) and the cable widths, which can be programmed in by the network operator when the modem is installed. The reference line background noise, $\tilde{\sigma}_k$, is set to the static line noise seen by the reference line. Finally, $\tilde{s}_k^n$ is the transmit power of user n's reference line on frequency tone k and s_k^n is the transmit power of user n on tone k.

In general, the concept of a reference line could be extended to multiple reference lines. For each M reference lines the optimization problem has up to

$M+1$ local maximums [9]. This increases the complexity but may also increase the performance.

The ASB optimization problem is now formulated as follows: each user n solves a different version of the following optimization problem:

$$\begin{aligned} \max_{s^n} \quad & R^{n,\text{ref}} \\ \text{subject to:} \quad & R^n \geq R^{n,\text{target}} \\ & \sum_k s_k^n \leq P^n \\ & 0 \leq s_k^n \leq s_k^{n,\text{mask}}, \quad \forall \quad k. \end{aligned} \tag{11}$$

For each user n, solving (11) requires optimizing its own PSD s_k^n which determines its own achievable rate (R^n) and the reference line ($R^{n,\text{ref}}$). After each user solves the optimization problem, (11), the crosstalk values of the network change. Each user must then remeasure or estimate the new crosstalk channel values and re-optimize using (11). This process is repeated until the PSD values of all users converge.

The approach to solving (11) is identical to those shown in Section 3.1 and Section 3.2 where the total number of users is two (user n and user n's reference line). The concept of dual decomposition is then applied as in the OSB and the ISB case giving:

$$\mathcal{L}_k^n = (1-\lambda^n)(w^n b_k^n + (1-w^n)\tilde{b}_k^n) - \lambda^n s_k^n.$$

The optimal PSD, $s_k^{n,*}$, is then,

$$s_k^{n,*} \triangleq \arg\max_{s_k^n \in [0,\, s_k^{n,\text{mask}}]} \mathcal{L}_k^n(w^n, \lambda^n, s_k^1, \ldots, s_k^N). \tag{12}$$

Equation (12) can be solved for by evaluating $\partial \mathcal{L}_k^n / \partial s_k^n = 0$. If $\alpha_k^{n,m}$ is defined as $\alpha_k^{n,m} \triangleq |h_k^{n,m}|^2 / |h_k^{n,n}|^2$ then Equation (12) is equivalent to

$$\frac{(1-\lambda^n)w^n}{s_k^{n,*} + \sum_{m \neq n} \alpha_k^{n,m} s_k^m + \sigma_k^n} - \frac{(1-\lambda^n)(1-w^n)\tilde{\alpha}_k^n \tilde{s}_k}{(\tilde{s_k} + \tilde{\alpha}_k^n s_k^{n,*} + \tilde{\sigma}_k)(\tilde{\alpha}_k^n s_k^{n,*} + \tilde{\sigma}_k)} - \lambda^n = 0. \tag{13}$$

Equation (13) can be simplified into a cubic equation in terms of $s_k^{n,*}$ and hence has three roots that can be solved for in closed-form. It is necessary to compare the value of $\mathcal{L}_k^n$ at each of the feasible roots as well as on the boundary conditions, $s_k^n = 0$ and $s_k^n = P^n$ (if no spectral mask is used, or $s_k^n = s_k^{n,\text{mask}}$ if one is used). The value of $s_k^{n,*}$ is chosen to be a feasible PSD corresponding to the maximum value of $\mathcal{L}_k^n$.

After solving for the optimal PSD value, user n updates λ^n to enforce the total power constraint and w^n to enforce the target rate constraint. The algorithm presented uses a simple bisection search to find the Lagrange multiplier values. Users iterate this process until all the users' PSDs converge. The complete algorithm is shown in Algorithm 3.

Algorithm 3: ASB Algorithm

```
Let $\epsilon_R > 0$ and $\epsilon_P > 0$ be given ;
Initialize PSDs: $s_k^n = P^n/K, \ \forall \ n, \ k$ ;
repeat
    foreach $n$ do
        Initialize $w_{\min}^n = 0, \ \ w_{\max}^n = 1$ ;
        while $|\sum_k b_k^n - R^{n,\text{target}}| > \epsilon_R$ do
            $w^n = (w_{\max}^n + w_{\min}^n)/2$ ;
            Initialize $\lambda_{\min}^n = 0, \ \ \lambda_{\max}^n = 1$ ;
            while $|\sum_k s_k^n - P^n| > \epsilon_P$ and $\sum_k s_k^n \leq P^n$ do
                $\lambda^n = (\lambda_{\max}^n + \lambda_{\min}^n)/2$ ;
                $s_k^n = \arg\max_{s_k^n \in [0, \ s_k^{n,\text{mask}}]} \mathcal{L}_k^n \ \ \forall \ k$ ;
                if $\sum_k s_k^n > P^n$ then
                    $\lambda_{\min}^n = \lambda^n$ ;
                end
                otherwise
                    $\lambda_{\max}^n = \lambda^n$ ;
                end
            end
            if $\sum_k b_k^n > R^{n,\text{target}}$ then
                $w_{\max}^n = w^n$ ;
            end
            otherwise
                $w_{\min}^n = w^n$ ;
            end
        end
    end
until $s_k^n$ converges $\forall \ k, \ n$ ;
```

3.4 Iterative Water-Filling (IWF)

The IWF algorithm [4] consists of performing water-filling on each user iteratively until convergence is achieved. Since the concept of water-filling is crucial in the IWF algorithm, it will be presented prior to discussing IWF itself.

Water-filling is the process by which one user (denoted here by $\tilde{n}$) attempts to maximize its rate capacity regardless of the effect on other users. More specifically, water-filling attempts to solve the following optimization problem:

$$\begin{aligned} \max_{\mathbf{s}^{\tilde{n}}} \quad & R^{\tilde{n}} \\ \text{subject to:} \quad & \sum_{k \in \mathcal{K}} s_k^{\tilde{n}} \leq P^{\tilde{n}}, \\ & 0 \leq s_k^{\tilde{n}} \leq s_k^{\tilde{n},\text{mask}}, \quad \forall \quad k. \end{aligned}$$

Since the maximization problem is concave, it can be solved by setting the derivative of its Lagrangian to 0. This results in the spectrum update formula with a single unknown, λ, shown in Equation (14).

$$\begin{aligned}\text{Define: } (s_k^n)^* &\triangleq \frac{1}{\lambda} - \frac{\sum_{m\neq 1} |h_k^{n,m}|^2 s_k^m + \sigma_k^n}{|h_k^{n,n}|^2/\Gamma}. \\ \text{Then} \quad s_k^n &= \begin{cases} s_k^{\tilde{n},\text{mask}} & \text{if } (s_k^n)^* > s_k^{\tilde{n},\text{mask}}, \\ 0 & \text{if } (s_k^n)^* < 0, \\ (s_k^n)^* & \text{otherwise.} \end{cases} \end{aligned} \tag{14}$$

From a maximization point of view, the power constraint should be active in order to achieve the highest rate. This can be achieved by varying λ. The water-filling algorithm using a bisection search to find λ is presented in Alogirthm (4).

Algorithm 4: Water-filling Algorithm for user n

```
Let $\epsilon > 0$ be given ;
Initialize PSD: $s_k = 0, \ \forall \ k$ ;
Initialize $\lambda_{\min} = 0, \ \lambda_{\max} = 1$ ;
Update PSD using (14) with $\lambda_{\max}$ as $\lambda$ ;
while $\sum_k s_k > P$ do
    $\lambda_{\max} = 2\lambda_{\max}$ ;
    Update PSD using (14) with $\lambda_{\max}$ as $\lambda$ ;
end
while 1 do
    $\lambda = (\lambda_{\max} + \lambda_{\min})/2$ ;
    Update PSD using (14) with $\lambda$ ;
    if $\sum_k s_k > P$ then
        $\lambda_{\min} = \lambda$ ;
    else if $P - \sum_k s_k \leq \epsilon$ then
        Break ;
    else
        $\lambda_{\max} = \lambda$ ;
    end
end
```

The IWF algorithm iteratively performs water-filling one user at a time. The users continuously perform water-filling in turn until a Nash Equilibrium point (a point where no user can benefit from changing its power allocation) is reached. When applying rate constraints, each user adjusts their water-filling process to ensure that they achieve their target rate. When applying water-filling, if a user achieves a rate higher than its target rate, that user's allowable total power is reduced; similarly, if a user achieves a rate lower than its target rate, that user's allowable total power is increased. Note that the allowable total power can never exceed the total power constraint. Therefore, the target rate set for the users must be feasible target rates for those users in order for the algorithm to converge. This process is summarized in Algorithm 5.

Algorithm 5: Iterative Water-filling Algorithm

```
Let $\epsilon > 0$ be given ;
Let $\Delta P > 0$ be given ;
Let $\Delta R > 0$ be given ;
Let $\tilde{P} = P \quad \forall\ n$ ;
Initialize PSD: $s_k^n = 0, \ \forall\ n,\ k$ ;
repeat
    foreach $n$ do
        Perform water-filling on user $n$ with $\tilde{P}$ as total allowable power ;
        if $R^n > R^{n,\text{target}}$ then
            $\tilde{P}^n = \tilde{P}^n - \Delta P$ ;
        else if $R^n < R^{n,\text{target}}$ then
            $\tilde{P}^n = \tilde{P}^n + \Delta P$ ;
        if $\tilde{P}^n > P^n$ then
            $\tilde{P}^n = P^n$ ;
        end
    end
until $s_k^n$ converges $\forall\ k,\ n$ ;
```

3.5 Selective Iterative Water-filling (SIW)

SIW [5] makes use of the fact that IWF forces users with better channels to restrict their allowable total power. At the same time, it acknowledges that each user is limited by a single water-filling level on each frequency tone. SIW attempts to remove these restrictions by allowing some users to perform another IWF process. More specifically, at the end of every IWF process, SIW allows users who did not use up all their transmit power to perform another IWF process without affecting the users who already used up their transmit powers. This allows some users to increase their water-filling level on some frequency tones, and does so without affecting other users which gives performance benefits over IWF. Therefore, SIW lets all users use up their allowable total power. The SIW algorithm is summarized in Algorithm 6.

The modified IWF used by SIW must guarantee that at least one user is using all of its allowable power. This is done to ensure that at each outer iteration of SIW, at least one user is removed from the new IWF process. One way of doing so is by maximizing one users rate while forcing all other users to meet their target rates.

At the end of every modified IWF process, the SIW algorithm looks for the users that have used up all of their allowable power and removes them from the next IWF round. As well, the frequency tones used by those users are also removed. The removal of the frequency tones is crucial in preserving the

Algorithm 6: Selective Iterative Water-filling Algorithm

```
Let $\tilde{\mathcal{N}} = \mathcal{N}$ ;
Let $\tilde{\mathcal{K}} = \mathcal{K}$ ;
Let $\tilde{P}^n = P^n \;\; \forall \;\; n$ ;
Let $n \in \tilde{\mathcal{N}}$ ;
while $\tilde{\mathcal{N}} \neq \emptyset$ and $\tilde{\mathcal{K}} \neq \emptyset$ do
    Perform modified IWF with $\tilde{\mathcal{N}}$, $\tilde{\mathcal{K}}$, and power constraints $\tilde{P}^n \; \forall \; n$ ;
    foreach $n \in \tilde{\mathcal{N}}$ do
        if $\sum_k s_k^n = P^n$ then
            $\tilde{\mathcal{N}} = \tilde{\mathcal{N}} \backslash \{n\}$ ;
            foreach $k \in \tilde{\mathcal{K}}$ do
                if $s_k^n \neq 0$ then
                    $\tilde{\mathcal{K}} = \tilde{\mathcal{K}} \backslash \{k\}$ ;
                end
            end
        end
    end
    foreach $n \in \mathcal{N}$ do
        $\tilde{P}^n = P^n - \sum_{k \in \mathcal{K} \backslash \tilde{\mathcal{K}}} s_k^n$ ;
    end
end
```

removed user's data rates. This process of performing IWF, removing users and frequency tones and re-performing IWF, is repeated until either all the frequency tones or all the users have been removed.

One fall-back of SIW is that it requires a messaging system to identify the removed users and tones.

3.6 Successive Convex Approximation for Low complExity (SCALE)

Successive Convex Approximation for Low complExity (SCALE) was first introduced in [10] and [11]. It solves problem (3) by approximating it with a concave lower bound, maximizing the approximation, and repeating the process with another approximation. SCALE introduces a method of distributing the required processing over multiple users.

SCALE uses the approximation $\alpha \log(z) + \beta \leq \log(1+z)$ with $\alpha \triangleq z_0/(1+z_0)$ and $\beta \triangleq \log(1+z_0) - z_0 \log(z_0)/(1+z_0)$. The inequality is tight when $z = z_0$. By applying the inequality to the total rate capacity equation and replacing the variables s_k^n with $\tilde{s}_k^n$ where $s_k^n = \exp(\tilde{s}_k^n)$, a concave equation is obtained yielding a concave maximization problem:

$$\max_{\tilde{\mathbf{s}}} \sum_k \sum_n w^n \alpha_k^n \log_2 \left(\frac{1}{\Gamma} \frac{|h_k^{n,n}|^2 \exp(\tilde{s}_k^n)}{\sum_{m \neq n} |h_k^{n,m}|^2 \exp(\tilde{s}_k^m) + \sigma_k^n} \right) + \beta_k^n,$$

which can be re-written as:

$$\max_{\tilde{\mathbf{s}}} \sum_k \sum_n w^n \alpha_k^n \quad \log_2 \left(\frac{|h_k^{n,n}|^2}{\Gamma} \right) + \tilde{s}_k^n - \log_2 \left(\sum_{m \neq n} |h_k^{n,m}|^2 e^{\tilde{s}_k^m} + \sigma_k^n \right) \Bigg) + \beta_k^n. \tag{15}$$

.

Any convex optimization tool can be used to solve Problem (15), but SCALE proposes a gradient approach that can also be distributed among each users by using the function's Lagrangian:

$$\mathcal{L}(\tilde{\mathbf{s}}, \boldsymbol{\lambda}) = -\sum_n \lambda^n \left(\sum_k e^{\tilde{s}_k^n} - P^n \right) +$$
$$\sum_k \sum_n w^n \alpha_k^n \quad \log_2 \left(\frac{|h_k^{n,n}|^2}{\Gamma} \right) + \tilde{s}_k^n - \log_2 \left(\sum_{m \neq n} |h_k^{n,m}|^2 e^{\tilde{s}_k^m} + \sigma_k^n \right) \Bigg) + \beta_k^n.$$

By taking the Lagrangian's gradient and setting it equal to zero, the spectrum update formula shown in Equation (16) is obtained:

$$s_k^n = \frac{w^n \alpha_k^n}{\lambda^n + \sum_{m \neq n} \frac{|h_k^{m,n}|^2 w^m \alpha_k^m}{\sum_{q \neq m} |h_k^{m,q}|^2 s_k^q + \sigma_k^m}}. \tag{16}$$

The denominator of the update equation, Equation (16), contains information about other users. This additional information allows the distributed algorithm to converge to a locally optimal point. In order to gather this information, a message passing system is required. Every user measures their total interference and noise on every tone and transmits it back to the Spectrum Management Center (SMC). Since the SMC has partial channel knowledge, the message passing system and update formula can be simplified as follows:

$$\begin{aligned} \mathcal{N}_k^n &= \frac{w^n \alpha_k^n}{\sum_{m \neq n} |h_k^{n,m}|^2 s_k^m + \sigma_k^n} \\ \mathcal{M}_k^n &= \sum_{m \neq n} |h_k^{m,n}|^2 \mathcal{N}_k^m \end{aligned}$$

$$s_k^n = \frac{w^n \alpha_k^n}{\lambda^n + \mathcal{M}_k^n}. \tag{17}$$

At every iteration, every user n calculates $\mathcal{N}_k^n$ on every tone and sends it to the SMC. The SMC produces the $\mathcal{M}_k^n$ values and distributes them to each user n. The full SCALE algorithm is summarized in Algorithm 7.

Algorithm 7: SCALE Algorithm

```
At each user $n$'s modem :
Initialize PSD: $s_k^n = 0, \ \forall \ k$ ;
Initialize $\alpha_k^n = 0, \ \forall \ k$ ;
repeat
    Receive $\mathcal{M}_k^n$ from SMC ;
    Update spectrum using (17) ;
    At every $m$ iterations, update $\alpha_k^n, \ \forall \ k$ ;
    Generate $\mathcal{N}_k^n$ and send to SMC ;
indefinitely

At the SMC :
repeat
    Receive $\mathcal{N}_k^n$ from every user ;
    Generate $\mathcal{M}_k^n$ and send to SMC ;
indefinitely
```

3.7 DSM using DCA

3.7.1 DC Programming and DCA Approach

This section introduces the definitions of DC programming and DCA and shows how Problem (3) can be reformulated as a DC optimization problem. DC programming and DC Algorithms (DCA) [15] [21] are tools used to solve a subset of non-convex optimization problems.

Definition 1. *A function $f(\mathbf{x})$ is a called a DC function with DC decomposition $g(\mathbf{x}) - h(\mathbf{x})$ if it can be written as:*

$$f(\mathbf{x}) = g(\mathbf{x}) - h(\mathbf{x}),$$

where $g(\mathbf{x})$ and $h(\mathbf{x})$ are lower semi-continuous proper convex functions on $\mathbb{R}^p$.

Definition 2. *The general DC programming problem is an optimization problem of the form:*

$$\inf_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in C \subseteq \mathbb{R}^p \}, \tag{P_{dc}}$$

where $f(\mathbf{x})$ is a DC function and C is a convex set.

The optimization problem in P_{dc} can be re-written by incorporating C into the objective function using an indicator function denoted by χ_C, as follows:

$$\inf_{\mathbf{x}} \{ \chi_C(\mathbf{x}) + g(\mathbf{x}) - h(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^p \}, \tag{18}$$

$$\text{where } \chi_C(\mathbf{x}) \triangleq \begin{cases} 0 & \text{if } \mathbf{x} \in C, \\ +\infty & \text{if } \mathbf{x} \notin C. \end{cases}$$

In order to construct the dual (D_{dc}) of the DC program (P_{dc}), first the conjugate function, g^*, of a convex function g, is defined.

Definition 3. *The conjugate function g^*, of a convex function g, is defined as follows:*

$$g^*(\mathbf{y}) \triangleq \sup_{\mathbf{y}} \{\langle \mathbf{x}, \mathbf{y} \rangle - g(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^p\}.$$

Then, the dual of the DC program (P_{dc}), can be written as:

$$\inf_{\mathbf{y}} \{h^*(\mathbf{y}) - g^*(\mathbf{y}) : \mathbf{y} \in \mathbb{R}^p\}. \tag{D_{dc}}$$

It is shown in Proposition 1 that the solution to the primal DC program (P_{dc}) and the dual DC program (D_{dc}) are equal.

Proposition 1. *The solution to the primal DC problem (P_{dc}) and the dual DC program (D_{dc}) are equal.*

Proof. Let α be the solution to the primal DC problem (P_{dc}). Therefore, $\alpha = \inf_{\mathbf{x}}\{g(\mathbf{x}) - h(\mathbf{x}) : \mathbf{x} \in C\ \}$.

By the definition of the conjugate function, shown in Definition 3,

$$\begin{aligned}
h^*(\mathbf{y}) &= \sup_{\mathbf{x} \in C} \{\langle \mathbf{x}, \mathbf{y} \rangle - h(\mathbf{x})\} \\
\Rightarrow h^*(\mathbf{y}) &\geq \langle \mathbf{x}, \mathbf{y} \rangle - h(\mathbf{x}) \ \ \forall\, \mathbf{x}, \mathbf{y} \\
\Rightarrow \ \ h(\mathbf{x}) &\geq \langle \mathbf{x}, \mathbf{y} \rangle - h^*(\mathbf{y}) \ \ \forall\, \mathbf{x}, \mathbf{y} \\
\Rightarrow \ \ h(\mathbf{x}) &= \sup_{\mathbf{y} \in C} \{\langle \mathbf{x}, \mathbf{y} \rangle - h^*(\mathbf{y})\}.
\end{aligned}$$

Note that by replacing $h(\mathbf{x})$ with $g(\mathbf{x})$, it follows that $g(\mathbf{x}) = \sup_{\mathbf{y} \in C} \{\langle \mathbf{x}, \mathbf{y} \rangle - g^*(\mathbf{y})\}$, hence:

$$\begin{aligned}
\alpha &= \inf_{\mathbf{x} \in C} \{g(\mathbf{x}) - h(\mathbf{x})\} \\
&= \inf_{\mathbf{x} \in C} \Big\{ g(\mathbf{x}) - \sup_{\mathbf{y} \in C} \{\langle \mathbf{x}, \mathbf{y} \rangle - h^*(\mathbf{y})\} \Big\} \\
&= \inf_{\mathbf{x} \in C} \left\{ \inf_{\mathbf{y} \in C} \Big\{ g(\mathbf{x}) - \{\langle \mathbf{x}, \mathbf{y} \rangle - h^*(\mathbf{y})\} \Big\} \right\} \\
&= \inf_{\mathbf{y} \in C} \ \beta(\mathbf{y}),
\end{aligned} \tag{19}$$

where $\beta(\mathbf{y}) \triangleq \inf_{\mathbf{x} \in C} \big\{ g(\mathbf{x}) - \{\langle \mathbf{x}, \mathbf{y} \rangle - h^*(\mathbf{y})\} \big\}$. It was shown in [15] that:

$$\beta(\mathbf{y}) = \begin{cases} h^*(\mathbf{y}) - g^*(\mathbf{y}) & \text{if } \mathbf{y} \in \operatorname{dom}\{h^*\}, \\ +\infty & \text{if } \mathbf{y} \notin \operatorname{dom}\{h^*\}. \end{cases}$$

Therefore, substituting $\beta(\mathbf{y})$ into Equation (19) gives:

$$\alpha = \inf_{\mathbf{y} \in \operatorname{dom}\{h^*\}} \big\{ h^*(\mathbf{y}) - g^*(\mathbf{y}) \big\}.$$

Therefore, α is also the solution to the dual problem (D_{dc}). $\square$

DCA is based on the local optimality conditions of (P_{dc}). In order to establish these conditions, the concept of a sub-differential is defined.

Definition 4. *The sub-differential of a convex function,* $\theta(\mathbf{x})$*, at a point* $\mathbf{x_0} \in \mathrm{dom}\{\theta(\mathbf{x})\} \triangleq \{\mathbf{x} \in \mathbb{R}^p : \theta(\mathbf{x}) < +\infty\}$*, is denoted by* $\partial\theta(\mathbf{x_0})$*, and is defined by:*

$$\partial\theta(\mathbf{x_0}) \triangleq \{\mathbf{y} \in \mathbb{R}^p : \theta(\mathbf{x}) \geq \theta(\mathbf{x_0}) + \langle \mathbf{x} - \mathbf{x_0}, \mathbf{y} \rangle \,,\ \forall\, \mathbf{x} \in \mathbb{R}^p\}.$$

The sub-differential generalizes the derivative in the sense that it is a set of derivatives at a given point, $\mathbf{x_0}$. More specifically, if the function θ were differentiable at $\mathbf{x_0}$, then $\partial\theta(\mathbf{x_0})$ would consist of only one value, the slope of θ at the point $\mathbf{x_0}$; however, if θ were not differentiable at $\mathbf{x_0}$ (e.g., a cusp or a corner point) then $\partial\theta(\mathbf{x_0})$ would consist of the set of all possible tangent lines which pass through the point $\mathbf{x_0}$. In fact, θ is differentiable at a point $\mathbf{x_0}$ if and only if $\partial\theta(\mathbf{x_0}) \equiv \nabla_{\mathbf{x}}\theta(\mathbf{x_0})$.

With the concept of sub-differentials, the local optimality conditions of (P_{dc}) at a point $\mathbf{x}^*$ (called a critical point of $f = g - h$), can be expressed as follows:

$$\partial h(\mathbf{x}^*) \cap \partial g(\mathbf{x}^*) \neq \emptyset, \tag{20}$$

$$\emptyset \neq \partial h(\mathbf{x}^*) \subset \partial g(\mathbf{x}^*). \tag{21}$$

Note that Equations (20) and (21) are necessary local optimality conditions for (P_{dc}), but for many classes of DC programs, they are also sufficient local optimality conditions [22] [21].

DCA iteratively minimizes the function $f = g - h$. On the r-th iteration, DCA approximates the function h by its affine minorization (i.e., taking $\mathbf{y}^r \in \partial h(\mathbf{x}^r)$) which forces f to be convex. DCA then minimizes the convex approximation using standard convex optimization techniques (i.e., determining a point $\mathbf{x}^{r+1} \in \partial g^*(\mathbf{y}^r)$). Proposition 2 shows that it is sufficient to find a point $\mathbf{x}^{r+1} \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle :\ \mathbf{x} \in \mathbb{R}^p\}$.

Proposition 2. $\mathbf{x}^* \in \partial g^*(\mathbf{y}^r) \iff \mathbf{x}^* \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle : \mathbf{x} \in \mathbb{R}^p\}$.

Proof. Let $x^* \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle :\ \mathbf{x} \in \mathbb{R}^p\}$.

Since $h(\mathbf{x}^r)$ is a constant, it can be removed from the minimization problem. As well, since

$$\langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle = \langle \mathbf{x}, \mathbf{y}^r \rangle - \langle \mathbf{x}^r, \mathbf{y}^r \rangle$$

where $\mathbf{x}^r$ and $\mathbf{y}^r$ are constants, it follows that $\langle \mathbf{x}^r, \mathbf{y}^r \rangle$ is also a constant and hence can also be removed from the minimization problem. Therefore,

$$\begin{aligned} \mathbf{x}^* &\in \arg\min_{\mathbf{x} \in \mathbb{R}^p}\{g(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle\} \\ \mathbf{x}^* &\in \arg\min_{\mathbf{x} \in \mathbb{R}^p}\{g(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y}^r \rangle\} \end{aligned}$$

which can be re-written as:

$$\begin{aligned} & g(\mathbf{x}^*) - \langle \mathbf{x}^*, \mathbf{y}^r \rangle \le g(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y}^r \rangle \,,\ \forall\ x \in \mathbb{R}^p \\ \Longleftrightarrow\quad & g(\mathbf{x}) \ge g(\mathbf{x}^*) + \langle \mathbf{x} - \mathbf{x}^*, \mathbf{y}^r \rangle \,,\ \forall\ x \in \mathbb{R}^p \\ \Longleftrightarrow\quad & \mathbf{y}^r \in \partial g(\mathbf{x}^*) \\ \Longleftrightarrow\quad & \mathbf{x}^* \in \partial g^*(\mathbf{y}^r). \end{aligned}$$

It follows that $x^* \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{x} - \mathbf{x}^r, \mathbf{y}^r \rangle : \ \mathbf{x} \in \mathbb{R}^p\} \Longleftrightarrow x^* \in \partial g^*(\mathbf{y}^r)$ since each step involves *if and only if* statements. □

Since $h(\mathbf{x}^r)$ and $\langle \mathbf{x}^r, \mathbf{y}^r \rangle$ are constants they can be removed from the minimization problem. Therefore, DCA must find a point:

$$\mathbf{x}^{r+1} \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y}^r \rangle : \mathbf{x} \in \mathbb{R}^p\}.$$

The DCA is outlined below in Algorithm 8. Note that calculating the arg min is a convex optimization problem and can be solved for using any standard convex optimization techniques.

Algorithm 8: General DCA

```
Initialize $\mathbf{x}^0$ to be a best guess ;
Initialize step count to $r = 0$ ;
repeat
    Calculate $\mathbf{y}^r \in \partial h(\mathbf{x}^r)$ ;
    Calculate $\mathbf{x}^{r+1} \in \arg\min_{\mathbf{x}}\{g(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y}^r \rangle : \mathbf{x} \in \mathbb{R}^p\}$ ;
    $r = r + 1$ ;
until $\mathbf{x}^r$ converges ;
```

Note that in order to properly apply DCA to solve (P_{dc}) one must first search for an appropriate DC decomposition for f, as well as an appropriate initial point. The choice of DC decomposition and initial point can have a crucial effect on both the performance and the run-time of the algorithm.

3.7.2 Applying DCA To The DSM Problem

The first step of applying DCA to the DSM problem is to re-write the RA optimization problem (3) in the form of a minimization problem,

$$\begin{aligned} \min_{\mathbf{s}^n,\ n \in \mathcal{N}} \quad & -\sum_{n \in \mathcal{N}} w^n R^n \\ \text{subject to:} \quad & \sum_{k \in \mathcal{K}} s_k^n \le P^n, \quad \forall \quad n \\ & 0 \le s_k^n \le s_k^{n,\text{mask}}, \quad \forall \quad n, k. \end{aligned} \tag{22}$$

The next step is to write the objective function in terms of the difference of two convex functions.

$$\begin{aligned}
-\sum_{n\in\mathcal{N}} w^n R^n &= -\sum_{n\in\mathcal{N}} w^n \Big(f_s \sum_{k\in\mathcal{K}} b_k^n \Big) \\
&= -\sum_{n\in\mathcal{N}} w^n \quad f_s \sum_{k\in\mathcal{K}} \log_2 \left(1 + \frac{1}{\Gamma} \frac{|h_k^{n,n}|^2 s_k^n}{\sum_{m\neq n} |h_k^{n,m}|^2 s_k^m + \sigma_k^n} \right) \Bigg) \\
&= -\sum_{n\in\mathcal{N}} w^n \quad f_s \sum_{k\in\mathcal{K}} \log_2 \left(\frac{\sum_{m\neq n} |h_k^{n,m}|^2 s_k^m + \big(|h_k^{n,n}|^2/\Gamma\big) s_k^n + \sigma_k^n}{\sum_{m\neq n} |h_k^{n,m}|^2 s_k^m + \sigma_k^n} \right) \Bigg).
\end{aligned}$$

For simplicity, define:

$$\begin{aligned}
\mathbf{s} &\triangleq \{s_k^n : n \in \mathcal{N}, k \in \mathcal{K}\} \\
H_k^{n,m} &\triangleq \begin{cases} |h_k^{n,n}|^2/\Gamma & \text{if } m = n \\ |h_k^{n,m}|^2 & \text{if } m \neq n \end{cases} \\
A_k^n(\mathbf{s}) &\triangleq \sum_{m\neq n} H_k^{n,m} s_k^m + \sigma_k^n \\
B_k^n(\mathbf{s}) &\triangleq A_k^n + H_k^{n,n} s_k^n.
\end{aligned}$$

The objective function of the DCA optimization problem (22) can then be re-written as:

$$\begin{aligned}
& -\sum_{n\in\mathcal{N}} w^n \quad f_s \sum_{k\in\mathcal{K}} \log_2 \left(\frac{B_k^n(\mathbf{s})}{A_k^n(\mathbf{s})} \right) \Bigg) \\
= \quad & -f_s \sum_{n,\,k} w^n \log_2 \Big(B_k^n(\mathbf{s}) \Big) \Bigg) - \quad -f_s \sum_{n,\,k} w^n \log_2 \Big(A_k^n(\mathbf{s}) \Big) \Bigg).
\end{aligned}$$

By defining:

$$\begin{aligned}
g(\mathbf{s}) &\triangleq -f_s \sum_{n,\,k} w^n \log_2 \Big(B_k^n(\mathbf{s}) \Big) \\
h(\mathbf{s}) &\triangleq -f_s \sum_{n,\,k} w^n \log_2 \Big(A_k^n(\mathbf{s}) \Big),
\end{aligned}$$

the objective function can be re-written as:

$$f(\mathbf{s}) = g(\mathbf{s}) - h(\mathbf{s}). \tag{23}$$

Using the fact that a summation of convex functions forms a convex function, and the fact that $-\log(\cdot)$ is convex, and that w^n and f_s are positive, it follows that $g(\mathbf{s})$ and $h(\mathbf{s})$ are also both convex functions. Therefore, $f(\mathbf{s}) = g(\mathbf{s}) - h(\mathbf{s})$

is a DC function. There are many different possible methods of applying DCA to solve this problem, one method is discussed in this section.

Any method for applying DCA requires that the derivatives of $g(\mathbf{s})$ and $h(\mathbf{s})$ are computed. For convenience, these derivatives are shown in Equations (24) and (25).

$$\frac{\partial g(\mathbf{s})}{\partial s_{\tilde{k}}^{\tilde{n}}} = -f_s \sum_{n} \frac{w^n}{\ln(2)} \frac{H_{\tilde{k}}^{n,\tilde{n}}}{A_k^n(\mathbf{s})} \tag{24}$$

$$\frac{\partial h(\mathbf{s})}{\partial s_{\tilde{k}}^{\tilde{n}}} = -f_s \sum_{n \neq \tilde{n}} \frac{w^n}{\ln(2)} \frac{H_{\tilde{k}}^{n,\tilde{n}}}{B_k^n(\mathbf{s})} \tag{25}$$

Note that the denominators of Equations (24) and (25) require evaluating N-dimensional and (N-1)-dimensional summations respectively.

Since the sub-problem for the DC decomposition ($f(\mathbf{s}) = g(\mathbf{s}) - h(\mathbf{s})$) is nonlinear, one method to solve it involves creating a new DC decomposition to simplify the sub-problem calculation. One DC decomposition (as shown in [15] and [23]) defines a new objective function as:

$$\tilde{f}(\mathbf{s}) \triangleq \tfrac{1}{2}\xi||\mathbf{s}||^2 - \left(\tfrac{1}{2}\xi||\mathbf{s}||^2 - f(\mathbf{s})\right),$$

where $\xi > 0$ to enforce convexity on $\tilde{g}(\mathbf{s}) \triangleq \frac{1}{2}\xi||\mathbf{s}||^2$. Based on the assumption that $\tilde{h}(\mathbf{s}) \triangleq \frac{1}{2}\xi||\mathbf{s}||^2 - f(\mathbf{s})$ is convex, applying DCA on $\tilde{f}(\mathbf{s}) = \tilde{g}(\mathbf{s}) - \tilde{h}(\mathbf{s})$ involves applying an affine minorization (multi-variant first order approximation) to $\tilde{h}(\mathbf{s})$, which avoids the original nonlinear optimization of $f(\mathbf{s})$.

The benefit of this DC decomposition (shown in [15] and [23]) is that it reduces the problem from a nonlinear optimization problem into a simple quadratic optimization problem, which is a very well-researched field containing many algorithms to find solutions. The first step is proving that there indeed exists a $\xi > 0$ such that $\tilde{h}$ is convex.

Theorem 1. *There exists a $\xi > 0$ such that the function*

$$\tilde{h}(\mathbf{s}) = \tfrac{1}{2}\xi||\mathbf{s}||^2 - f(\mathbf{s})$$

is convex on the set $C \triangleq \{\mathbf{s} \in \mathbb{R}^{K\times N} \ : \ \sum_k s_k^n \leq P^n, \ 0 \leq s_k^n \leq s_k^{n,mask} \quad \forall \ n, \ k\}$.

Proof. Note that $\tilde{h}(\mathbf{s})$ is twice differentiable and its Hessian is $\nabla^2\tilde{h}(\mathbf{s}) = \xi I - \nabla^2 f(\mathbf{s})$ with $f(\mathbf{s}) \triangleq g(\mathbf{s}) - h(\mathbf{s})$ where $g(\mathbf{s})$ and $h(\mathbf{s})$ are convex on C. Recall that a function is convex on C if and only if its Hessian is positive semi-definite on C. Therefore, it is sufficient to show that $\nabla^2\tilde{h}(\mathbf{s})$ is positive semi-definite to show that $\tilde{h}(\mathbf{s})$ is a convex function on C. Therefore,

$$\begin{aligned}\nabla^2\tilde{h}(\mathbf{s}) &= \xi I - \nabla^2 f(\mathbf{s}) \\ &= \xi I - \nabla^2 g(\mathbf{s}) + \nabla^2 h(\mathbf{s}).\end{aligned}$$

Since $g(\mathbf{s})$ and $h(\mathbf{s})$ are convex, $\nabla^2 g(\mathbf{s})$ and $\nabla^2 h(\mathbf{s})$ are positive semi-definite. In order to show that $\nabla^2\tilde{h}(\mathbf{s})$ is positive semi-definite, it suffices to show that

$\xi I - \nabla^2 g(\mathbf{s})$ is positive semi-definite since summations of positive semi-definite matrices are positive semi-definite.

Recall that by the real spectral theorem, a real matrix A is symmetric if and only if there exists a real orthogonal matrix Q such that $A = Q^T \Lambda Q$, where Λ is a diagonal matrix. As well, recall that a real orthogonal matrix Q is defined as $Q^T Q = QQ^T = I$.

Since $g(\mathbf{s})$ is continuous, $\nabla^2 g(\mathbf{s})$ is symmetric. Therefore there exists a real orthogonal matrix Q such that $\nabla^2 g(\mathbf{s}) = Q^T \Lambda Q$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ and λ_i is the i-th eigenvalue of $\nabla^2 g(\mathbf{s})$.

Therefore, $\xi I - \nabla^2 g(\mathbf{s})$ can be re-written as:

$$\begin{aligned}
\xi I - \nabla^2 g(\mathbf{s}) &= \xi I - Q^T \Lambda Q \\
&= Q^T \Big((Q^T)^{-1} (\xi I) Q^{-1} \Big) Q - Q^T \Lambda Q \\
&= Q^T \Big(\xi (Q^T)^{-1} Q^{-1} - \Lambda \Big) Q \\
&= Q^T \Big(\xi (Q^{-1})^{-1} Q^{-1} - \Lambda \Big) Q \\
&= Q^T \big(\xi - \Lambda \big) Q \\
&= Q^T \Psi Q,
\end{aligned}$$

where $\Psi \triangleq \text{diag}(\xi - \lambda_1, \ldots, \xi - \lambda_N)$.

Therefore, the eigenvalues of the matrix $\xi I - \nabla^2 g(\mathbf{s})$ are $\xi - \lambda_1, \ldots, \xi - \lambda_N$. Recall that a matrix is positive definite if all its eigenvalues are strictly positive. As well, every positive definite matrix is also positive semi-definite. Therefore, if $\xi > \max\{\lambda_1, \ldots, \lambda_N\}$, all the eigenvalues of the matrix $\xi I - \nabla^2 g(\mathbf{s})$ will be strictly positive and therefore it will be positive definite, as well as positive semi-definite.

As previously stated, summations of positive semi-definite matrices are positive definite. Therefore,

$$\nabla^2 \tilde{h}(\mathbf{s}) = \Big(\xi I - \nabla^2 g(\mathbf{s}) \Big) + \nabla^2 h(\mathbf{s})$$

will also positive semi-definite.

Therefore, if $\xi > \max\{\lambda_1, \ldots, \lambda_N\}$, $\tilde{h}(\mathbf{s})$ will be convex on the set C. $\square$

Corollary 1. *If $\xi > \|\nabla^2 g(\mathbf{s})\|_\infty$, the function $\tilde{h}(\mathbf{s}) = \frac{1}{2}\xi ||\mathbf{s}||^2 - f(\mathbf{s})$ will be convex on the set $C \triangleq \{\mathbf{s} \in \mathbb{R}^{K \times N} : \sum_k s_k^n \leq P^n,\ 0 \leq s_k^n \leq s_k^{n,mask} \quad \forall\ n,\ k\}$.*

Proof. Recall that if v is an eigenvector of matrix A with eigenvalue λ:

- $\|A\|_\infty = \max_i \sum_j |a_{ij}|$, which is the maximum absolute row sum of the matrix.
- $\|Av\|_\infty \leq \|A\|_\infty \cdot \|v\|_\infty$
- $Av = \lambda v$

- $\|\lambda v\|_\infty = |\lambda| \cdot \|v\|_\infty$.

Let v_{max} be the corresponding eigenvector for the maximum eigenvalue, λ_{max}, of the matrix $\nabla^2 g(\mathbf{s})$. Therefore,

$$\begin{aligned}
&\|\nabla^2 g(\mathbf{s}) v_{max}\|_\infty \leq \|\nabla^2 g(\mathbf{s})\|_\infty \cdot \|v_{max}\|_\infty \\
&\Rightarrow \|\ \lambda_{max} v_{max}\|_\infty \leq \|\nabla^2 g(\mathbf{s})\|_\infty \cdot \|v_{max}\|_\infty \\
&\Rightarrow |\lambda_{max}| \cdot \|\ v_{max}\|_\infty \leq \|\nabla^2 g(\mathbf{s})\|_\infty \cdot \|v_{max}\|_\infty \\
&\Rightarrow |\lambda_{max}| \leq \|\nabla^2 g(\mathbf{s})\|_\infty.
\end{aligned}$$

By Theorem 1, if $\xi > \max\{\lambda_1, \ldots, \lambda_N\}$, $\tilde{h}(\mathbf{s})$ will be convex on the set C. Therefore, by choosing

$$\xi > \|\nabla^2 g(\mathbf{s})\|_\infty \geq |\lambda_{max}| \geq \max\{\lambda_1, \ldots, \lambda_N\},$$

and applying Theorem 1, $\tilde{h}(\mathbf{s})$ will be convex on the set C. □

Theorem 1 shows that there exists a $\xi > 0$, such that $\tilde{f}(\mathbf{s}) = \tilde{g}(\mathbf{s}) - \tilde{h}(\mathbf{s})$ is DC, where $\tilde{g}(\mathbf{s}) \triangleq \frac{1}{2}\xi||\mathbf{s}||^2$, $\tilde{h} \triangleq \frac{1}{2}\xi||\mathbf{s}||^2 - f(\mathbf{s})$, and $f(\mathbf{s})$ is the original objective function defined in Equation (23). Corollary 1 provides a method for finding such a ξ.

Therefore, DCA can be applied to $\tilde{f}(\mathbf{s})$. Starting from an initial point, $\mathbf{s}^{(0)} \in C$, two sequences $\{\mathbf{s}^{(k)}\}$ and $\{\mathbf{q}^{(k)}\}$ are constructed such that, $\mathbf{q}^{(k)} \in \partial\tilde{h}(\mathbf{s}^{(k)})$ and $\mathbf{s}^{(k+1)} \in \partial(\tilde{g} + \chi_C)^*(\mathbf{q}^{(k)})$.

As shown in Section 3.7.1,

$$\mathbf{s}^{(k+1)} \in \partial(\tilde{g} + \chi_C)^*(\mathbf{q}^{(k)}) \iff \mathbf{s}^{(k+1)} \in \arg\min_{\mathbf{s}} \tilde{g}(\mathbf{s}) - \langle \mathbf{s}, \mathbf{q}^k \rangle : \mathbf{s} \in C.$$

The algorithm for applying this method of DCA to DSM is shown in Algorithm 9.

Algorithm 9: DCA For DSM

```
Initialize $\mathbf{s}^{(0)}$ to be a best guess ;
Initialize step count to $r = 0$ ;
repeat
    Calculate $\mathbf{q}^r = \nabla\tilde{h}(\mathbf{s}^{(r)}) = \xi\mathbf{s}^{(r)} - \nabla\tilde{f}(\mathbf{s}^{(r)})$ ;
    Calculate $\mathbf{s}^{r+1} = \arg\min_{\mathbf{s}}\{\frac{1}{2}\xi||\mathbf{s}||^2 - \langle \mathbf{s}, \mathbf{q}^{(r)} \rangle : \mathbf{s} \in C\}$ ;
    $r = r + 1$ ;
until $\mathbf{s}^r$ converges ;
```

Using the fact that the DC component, $\tilde{h}$, is differentiable, and discussions in [15], [24], DCA has the following convergence properties:

(i) DCA is a descent method (i.e., at each step of the algorithm the objective function value decreases).

(ii) If the optimal value of the primal problem (P_{dc}) is finite, and the sequences $\{\mathbf{s}^r\}$ and $\{\mathbf{q}^r\}$ are bounded, then all limit points $\mathbf{s}^*$ and $\mathbf{q}^*$ of the sequences $\{\mathbf{s}\}$ and $\{\mathbf{q}\}$ are critical points of $g(\mathbf{s})-h(\mathbf{s})$ and $h^*(\mathbf{q})-g^*(\mathbf{q})$ respectively (i.e., $\partial h(\mathbf{s}^*) \cap \partial g(\mathbf{s}^*) \neq \emptyset$ and $\partial h(\mathbf{q}^*) \cap \partial g(\mathbf{q}^*) \neq \emptyset$).

(iii) DCA has worst-case linear convergence for DC programs.

Property (i) allows for situational trade-offs between run-time and performance (i.e., pre-maturely terminating the algorithm in order to ensure faster run-time) while still ensuring that the objective function value of the found point is lower than that of the starting point. Property (ii) shows that based on the given assumptions, DCA will find a locally optimal solution. Property (iii) specifies the theoretical worst-case convergence time of DCA for a specific class of optimization problems, in practice the convergence time is often quicker.

4 Illustrative Examples

This section will discuss the performance (achievable data rate) for various DSM algorithms in a typical near-far deployment. The test case used is a non-symmetric frequency-selective network topology consisting of two users (lines). It has been shown that in symmetric or frequency-flat environments, many DSM algorithms perform near-optimally. Therefore the non-symmetric frequency selective near-far deployment, which is more common in practice, is of interest.

The test case considers an upstream transmission. The first line (user 1) runs 1500 ft from customer one to the central office (CO), and the second line (user 2) runs 3000 ft from customer two to the CO, as shown in Figure 1. The transmitting modems (TX) are located at the customers' homes and the receivers (RX) are located at the CO. Each DSL modem transmits over multiple frequency tones.

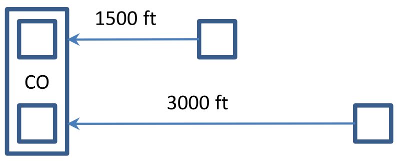

Figure 1: Network Topology.

In order to discuss the performance of different DSM algorithms, a rate region is a plotted which shows all the possible achievable data rate combinations that can be achieved. In general, for N users this would be an N-dimensional plot. Therefore the two-user case allows for easier visualization. The concepts extend naturally to the multi-user case.

Points operating on the boundary of the rate region are considered to be optimal points. The goal of DSM algorithms is to generate points which are closest to the optimal values, while operating at a low computational complexity.

The test case represents a realistic upstream VDSL scenario [18]. The test case assumes 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 Discrete Multi-Tone (DMT) symbol rate is $f_s = 4$ kHz. The FDD band plan 998 [25] is used, which consists of two separate upstream transmission bands: 3.75 – 5.2 MHz and 8.5 – 12 MHz. The optional 30 – 138 kHz band is not used. A maximum transmit power of 11.5 dBm is applied to each modem for each DSM algorithm.

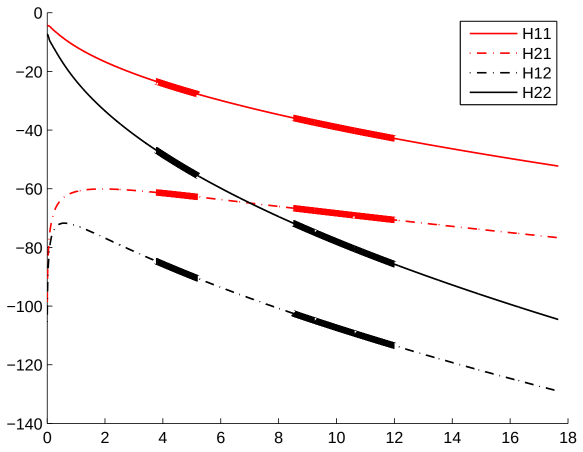

Figure 2: Two User Test Case.

The direct and the Far-End crossTalk (FEXT) transfer functions for the two-user test case are shown in Figure 2. User 1 represents the 1500 ft line and user 2 represents the 3000 ft line. Therefore, H11 and H22 are user 1 and user 2's direct transfer functions, respectively. As well, H21 is the FEXT transfer function from user 1 to user 2, and H12 is the FEXT transfer function from user 2 to user 1. Figure 2 shows the transfer function over all 4096 frequency tones; specifically, the bolded parts of the curves are the upstream bands used for DSM.

Simulations were run for: OSB, ISB, ASB, IWF, SIW, SCALE, DCA, and Flat PBO. The results are summarized in Figure 3. It is important to note that because of the simplicity of the two user channel, without the presence of background noise, the rate regions are more optimistic than usual. The test case used has few locally optimal points and therefore SCALE and DCA are able to converge to globally optimal points. In general, this is not the case, as with more interfering users the number of locally optimal points grows significantly, which leads to reduced performance in locally optimizing algorithms (e.g., DCA and SCALE).

Flat PBO is a SSM technique and therefore, it is evident that the performance benefits of DSM are significant. In more complicated channels, where the number of users increase, the performance of Flat PBO (and all SSM techniques) is far worse.

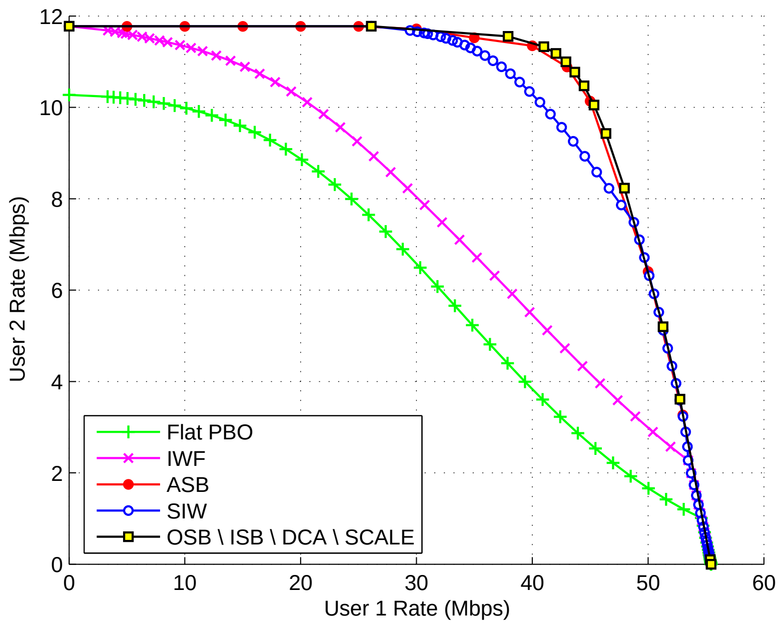

Figure 3: Two User Rate Regions.

It is important to note that while the rate curve allows for easy visualization, another significant factor in choosing a DSM algorithm is the run-time. The run-time of different algorithms depends on many factors including the processor(s) used, the coding used, the programming language used, etc. Therefore, the other performance metric used to compare the DSM algorithms will be their computational complexity. A comparison of the computational complexities of the various DSM algorithms is shown in Section 5.

5 Complexity

Computational complexity represents the number of operations required for the execution of an algorithm. More specifically, it shows the behavior of different algorithms as the input sizes increase. In order to represent this, $\mathcal{O}(\cdot)$ notation is used. It describes the limiting behavior of a function when the arguments tend towards infinity. This is described mathematically as follows:

$$f(\mathbf{x}) = \mathcal{O}(g(\mathbf{x})) \Longleftrightarrow |f(\mathbf{x})| \leq c|g(\mathbf{x})| \quad \forall \quad \mathbf{x} \quad \text{as} \quad \mathbf{x} \to \infty,$$

where $c \in \mathbb{R}$ is some constant.

Therefore, for an arbitrary algorithm dependent on some variable X, a $\mathcal{O}(X)$ algorithm will converge faster than a $\mathcal{O}(X^2)$ algorithm for sufficiently large X. The convergence time of an algorithm is said to be exponential if it is $\mathcal{O}(e^X)$, and is said to be polynomial if it is $\mathcal{O}(X^a)$ for $a > 1$. It is important to note that in algorithmic terms, exponential time is extremely slow.

For DSM purposes, there are two main inputs (variables) of interest: the number of users, N, and the number of frequency tones, K. Hence, all algorithms will be of the form $\mathcal{O}(f(N, K))$ for some function $f(N, K)$. The total power and target data rate constraints, shown in Equation (2), cause coupling over the frequency tones, which requires that each user's PSD be searched for jointly over all frequency tones.

Solving an X-dimensional exhaustive search has a computational complexity of $\mathcal{O}(e^X)$. Therefore, solving the DSM optimal resource allocation through an exhaustive search in both the number of users, N, and the number of frequency tones, K, would lead to a computational complexity of $\mathcal{O}(e^{NK})$, which is computationally intractable.

In Section 1, the concept of centralized versus distributed (and autonomous) systems was introduced. One key difference is in how the total interference plus noise is found. For autonomous algorithms, the total interference plus noise is measured locally by each user. For centralized algorithms, since the DSL channel is generally time-invariant, channel measurements can be taken a priori and stored at the Central Office (CO), these measurements would need to be updated periodically. Hence, full channel knowledge is assumed (i.e., $|h_k^{n,m}|^2$) is known $\forall\ n, m \in \mathcal{N},\ k \in \mathcal{K}$). Depending on the implementation, distributed algorithms can either operate fully autonomously or in a centralized manner by making use of explicit message passing.

Operating in a centralized manner may require the time-consuming computation of the interference plus noise for each user on each frequency tone in the system. More specifically, to compute the total interference plus noise for some user n on a given frequency tone k, one must cycle through all the users $m \neq n$ and multiply the channel transfer function by user m's transmit PSD for the specific frequency tone k and sum them. Hence, in order to compute the total interference plus noise for all N users on some frequency tone k, the total complexity will be:

$$N \times (N - 1) = N^2 - N.$$

Therefore, the computational complexity of calculating the total interference plus noise for all N users on some frequency tones k will be:

$$C_{\text{Interference}} = \mathcal{O}(N^2).$$

In addition, computing the bit-rates of all users N on some frequency tone k is dominated by computing the total interference plus noise and therefore, is also $\mathcal{O}(N^2)$. Note that in order to compute the bit-rates of all users N on all frequency tones K, the computational complexity will be $\mathcal{O}(KN^2)$.

5.1 IWF Complexity

Iterative Water-Filling (IWF) is a distributed algorithm briefly described in Section 3.4 (Algorithm 5). IWF consists of two loops where the outer loop cycles through the inner loop for all users. The inner loop performs water-filling for the n-th user over all the K frequency tones. The water-filling process involves a bisection search for that user's Lagrange multiplier, λ; Assuming that the number of bisection iterations required for the PSD to converge is at most v_1 and assuming that the number of iterations required for the IWF algorithm to converge is v_2, the total complexity of the IWF algorithm will be $v_1 \times v_2 \times N \times K$, which implies:

$$C_{IWF} = \mathcal{O}(KN).$$

It is important to note that for IWF, anytime a user is required to perform water-filling, that user must re-measure its total interference plus noise. Therefore, for N users and v iterations, a total of vN measurements must be performed. If the channel conditions are poor, the total number of iterations, v, may be large. Therefore, if the values of N and v are large, taking vN measurements may be a bottleneck in terms of run-time.

For such situations, IWF can be implemented in a centralized manner. As explained previously in Section 5, since the DSL channel is very slow time-varying, channel measurements can be taken a priori and stored at the CO. The CO can then compute the total interference plus noise seen by user n in a centralized manner, which requires $\mathcal{O}(KN)$ operations. The CO runs a virtual IWF game to generate the new PSD values. More specifically, on user n's turn in the virtual game, the CO computes the total interference plus noise seen by user n and then updates user n's virtual PSD. The CO plays this virtual game until all of the users' PSDs converge, at which point the CO informs each user of their new PSD value.

Implementing IWF in a centralized manner, using the same definitions as for the distributed manner, would require a total complexity of $N \times (KN + v_1 \times v_2 \times K)$, which implies:

$$C_{IWF}(\text{centralized}) = \mathcal{O}(KN^2).$$

Note that $C_{IWF}(\text{centralized})$ has a higher complexity than C_{IWF} since it performs more computations during every iteration. In practice, the run-time between the two methods can vary greatly depending on the number of users in the system, as well as the channel conditions, which both affect the number of iterations and channel measurements required.

5.2 SIW Complexity

The Selective Iterative Water-filling (SIW) algorithm (described in Section 3.5, Algorithm 6) performs two loops where the outer loop cycles through the inner loop for all users where the inner loop performs IWF for all users. At every iteration of the outer loop, users (and their corresponding frequency tones) that have already achieved their maximum power are removed. It has been shown that each sequential inner loop removes at least one user and frequency tone. This process is repeated until either all the users have been removed from the inner loop (maximum of N iterations), or until all frequency tones have been removed (possibly in less than N iterations). Therefore, the maximum number of additional outer loop iterations is N, which leads to a total complexity of up to N times that of IWF, which implies:

$$C_{SIW} = \mathcal{O}(KN^2).$$

5.3 OSB Complexity

The Optimal Spectrum Balancing (OSB) Algorithm is described in Section 3.1 (Algorithm 1). Assuming that the number of discrete bit-loadings is X, then the search space for all user's PSDs will contain X^N elements. Therefore, to solve for the optimal PSD for each user, an exhaustive search must be performed over the X^N combinations. For each bit-loading combination, the total rate must be calculated, which adds a multiplicative factor of N^2.

The OSB algorithm performs the above exhaustive search independently on each of the K frequency tones, and repeats itself until each user's Lagrange multipliers converge. Assuming that the number of iterations until all the Lagrange multipliers converge is v_1, the total complexity of the OSB algorithm is $v_1 \times K \times N^2 \times X^N$, which implies:

$$C_{OSB} = \mathcal{O}(KN^2X^N).$$

5.4 ISB Complexity

The Iterative Spectrum Balancing (ISB) algorithm is described in Section 3.2 (Algorithm 2). Assuming that the number of discrete bit-loadings is X, then the search space for each user's PSD will contain X elements. For ISB, there are two loops where the outer loop cycles through the inner loop for all users. The inner loop performs N different 1-dimensional exhaustive searches over each of the X elements for each user. For each bit-loading combination, the total rate must be calculated, which adds a multiplicative factor of N^2. Therefore, with respect to the number of users, N, each inner loop consists of $N^2 + N$ calculations. Assuming that the number of iterations required for the user's PSDs to converge is v_1, and that the number of iterations for w^n and λ^n to converge is v_2, the total complexity of the ISB algorithm is $v_1 \times v_2 \times K \times (N^2 + N) \times N \times X$, which implies:

$$C_{ISB} = \mathcal{O}(KN^3X).$$

5.5 ASB Complexity

The Autonomous Spectrum Balancing (ASB) algorithm is described in Section 3.3 (Algorithm 3). ASB has three nested iterations; the outermost cycle iterates through the users. Within each cycle, each user runs an outer loop which updates w^n until the target data rate is achieved, and an inner loop which updates λ^n until the power constraint is satisfied. Bisection search is used in both loops and therefore, the number of iterations required for both the outer and inner loops to converge, while achieving an accuracy of ϵ_R and ϵ_P respectively is $\log_2(1/\epsilon_R) \times \log_2(1/\epsilon_P)$.

The complexity of each iteration is dominated by finding the roots of the cubic polynomial, solving equation (13). Finding the roots of a cubic equation requires 44 operations in total [26]. This equation must be solved for each of the K frequency tones, therefore the total complexity of finding the roots is $44K$. Therefore, assuming that the algorithm converges in a finite number of iterations, v, the total complexity of the ASB algorithm is $v \times N \times \log_2(1/\epsilon_R) \times \log_2(1/\epsilon_P) \times 44K$ which implies:

$$C_{ASB} = \mathcal{O}(NK).$$

5.6 SCALE Complexity

The Successive Convex Approximation for Low complExity (SCALE) algorithm is described in Section 3.6 (Algorithm 7). SCALE is an iterative algorithm, each iteration begins by user n updating its PSD on all K frequency tones. Until the Lagrange multipliers for each user converge, the process of measuring their total interference plus noise, sending the measurements to the Spectrum Management Center (SMC), waiting for a reply and finally updating values accordingly, is repeated. Assuming v is the number of operations required to update a given user's PSD, the total complexity from a distributed point of view is vK.

The SMC receives a message constructed from the measured total interference plus noise from all users. The SMC then sends each user the sum of the messages received from all the other users. Therefore, generating the message for each user requires $K \times N$ additions. This process must be repeated for all N users, leading to a complexity of KN^2 at the SMC.

Therefore, the total complexity of the SCALE algorithm is $KN^2 + vK$, which implies:

$$C_{SCALE} = \mathcal{O}(KN^2).$$

5.7 DCA Complexity

Section 3.7.1 discusses several possible approaches to solving the DSM problem using Difference of Convex function Algorithms (DCA). There are an infinite amount of possible ways to formulate the problem as a difference of convex functions. For any reasonable decomposition, the complexity will be dominated by calculating the gradient of the objective function and applying convex optimization techniques. Since there are KN variables (e.g., s_k^n for some k and some n), calculating the gradient requires taking KN derivatives. For each of the KN derivatives, as shown in Equation (24) (respectively Equation (25)), a nested summation over all N elements (respectively $N-1$ elements) must be calculated. This requires $KN \times N^2$ computations the first time it is computed.

By noticing that the denominator in Equation (24) (respectively, (25)) is the same for all KN users, additional computations can be avoided by re-using the previously computed summation. For each additional user the number of computations required will be $KN \times N$. Therefore the complexity of taking the derivative will be:

$$C_{DCA}(\text{Derivative}) = \mathcal{O}(KN^3 - KN^2) = \mathcal{O}(KN^3)$$

at each iteration. Therefore, assuming that the number of iterations required until convergence is achieved is v, the total complexity is $v \times (KN^3 - KN^2 + \text{Solving Convex Optimization Problem})$, which implies:

$$C_{DCA} = \mathcal{O}(KN^3 + \text{Solving Convex Optimization Problem}).$$

6 Concluding Remarks

This technical report presented the concept of DSM and discussed its advantages in comparison to SSM. The following DSM algorithms were discussed: OSB, ISB, ASB, IWF, SIW, SCALE and one approach to solving the DSM problem using DCA. Through the use of an illustrative example (Section 4), it was shown that DCA was comparable in terms of performance (achievable data rate) to existing state-of-the-art DSM algorithms. Through the use of computational complexity analysis (Section 5), it was shown that DCA would be computationally tractable for large numbers of users and frequency tones. Therefore, the DCA approach discussed is competitive with current state-of-the-art DSM algorithms, both in terms of the achievable data rate and the computational complexity.

DCA and SCALE iteratively apply an approximation for the objective function and make use of the well-researched field of convex optimization to solve it. Based on the complexity analysis shown in Section 5, SCALE has a lower complexity than DCA, but more thorough testing in different multi-user (more practical) settings need to be conducted to compare the achievable data rates of both algorithms, since the DCA method possesses many heuristics which can have significant effects on both the performance and run-time.

Acknowledgements The authors would like to thanks Prof. Hoai-An Le-Thi and Mr. Anh Son Ta of Université Paul Verlaine (Metz, France) for the great discussions on DCA. Christopher Leung would also like to thank Mr. Duc Quynh Tran and Mr. Thuan Quang Nguyen for their help with DCA.

References

[1] Broadband Forum, "3 million new north-american broadband subscribers added in first quarter 2009 despite economic downturn," *News Release.*, June 16, 2009.

[2] ANSI Standard T1.417-2001, "Spectrum management for loop transmission systems," 2001.

[3] T. Starr, M. Sorbara, J. M. Cioffi, and P. J. Silverman, *DSL Advances*. Prentice-Hall, 2003.

[4] W. Yu, G. Ginis, and J. Cioffi, "Distributed multiuser power control for digital subscriber lines," *IEEE J. Select. Areas in Commun.*, vol. 20, pp. 1105–1115, June 2002.

[5] Y. Xu, S. Panigrahi, and T. Le-Ngoc, "Selective iterative water-filling for digital subscriber lines (DSL)," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, Article ID 59068, 11 pages doi:10.1155/2007/59068, 2007.

[6] R. Cendrillon, W. Yu, M. Moonen, J. Verlinden, and T. Bostoen, "Optimal multiuser spectrum management for digital subscriber lines," *IEEE Trans. Commun.*, vol. 54, pp. 922–933, May 2006.

[7] R. Cendrillon and M. Moonen, "Iterative spectrum balancing for digital subscriber lines," *Proc. IEEE Int. Conf. Commun.*, May 2005.

[8] R. Lui and W. Yu, "Low-complexity near-optimal spectrum balancing for digital subscriber lines," *Proc. IEEE Int. Conf. Commun.*, May 2005.

[9] R. Cendrillon, J. W. Huang, M. Chiang, and M. Moonen, "Autonomous spectrum balancing for digital subscriber lines," *IEEE Trans. Signal Process.*, vol. 55, pp. 4241–4257, Aug. 2007.

[10] J. Papandriopoulos and J. S. Evans, "Low-complexity distributed alogirthms for spectrum balancing in multi-user DSL networks," *Proc. IEEE Int. Conf. Commun.*, June 2006.

[11] J. Papandriopoulos and J. S. Evans, "Scale: A low-complexity distributed protocol for spectrum balancing in multiuser DSL networks," *IEEE Trans. Inform. Theory*, vol. 55, pp. 3711–3724, Aug. 2009.

[12] T. Hoang, "Convex analysis and global optimization," *Kluwer*, 1998.

[13] Y. Xu, T. Le-Ngoc, and S. Panigrahi, "Global concave minimization for optimal spectrum balancing in multi-user DSL networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 2875–2885, July 2008.

[14] R. Horst, T. Q. Phong, N. V. Thoai, and J. de Vries, "On solving a D.C. programming problem by a sequence of linear programs," *Journal of Global Optimization*, vol. 1, pp. 183–204, 1991.

[15] D. T. Pham and H. A. Le Thi, "Convex analysis approach to d.c. programming: Theory, algorithms and applications," *Acta Mathematica Vietnamica*, vol. 22, pp. 289–355, 1997.

[16] T. Starr, J. Cioffi, and P. Silverman, *Understanding Digital Subscriber Line Technology.* Upper Saddle River, NJ: Prentice-Hall, 1999.

[17] A. Forouzan and L. Garth, "Generalized iterative spectrum balancing and grouped vectoring for maximal throughput of digital subscriber lines," *Proc. IEEE Global Telecommun. Conf.*, vol. 4, pp. 2359–2363, May 2005.

[18] ETSI, "Transmission and multiplexing (TM); access transmission systems on metallic access cables; very high speed digital subscriber line (VDSL); part 1: Functional requirements," *ETSI Std. TS 101 270-1*, vol. Rev. V. 1.3.1, 2003.

[19] "Very high speed digital subscriber line (VDSL) metalic interface," *ANSI Std.*, vol. T1.424, 2004.

[20] "Very high speed digital subscriber line transceivers 2," *ITU Draft Std.*, vol. G.993.2, 2006.

[21] H. A. Le Thi and D. T. Pham, "The DC programming and DCA revisted with DC models of real world non-convex optimization problems," *Annals of Operations Research*, vol. 133, pp. 23–46, 2005.

[22] H. A. Le Thi and D. T. Pham, "Solving a class of linearly constrained indefinite quadratic problems by dc algorithms," *Journal of Global Optimization*, vol. 11, No 3, pp. 253–285, 1997.

[23] R. Blanquero and E. Carrizosa, "On covering methods for D.C. optimization," *Journal of Global Optimization*, vol. 18, pp. 265–274, 2000.

[24] D. T. Pham and H. A. Le Thi, "Dc optimization algorithms for solving the trust region subproblem," *SIAM J. Optimization*, vol. 8, pp. 476–505, 1998.

[25] K. McCammon, "G. vdsl: VDSL band plan for north america," *ITU contribution D*, vol. 715, 2000.

[26] R. Nickalls, "A new approach to solving the cubic: Cardan's solution revealed," *Mathematical Gazette*, vol. 77, pp. 354–359, 1993.