Bitloading with reduced computational complexity order for a multicarrier multimode PON

R.O. Taniman [1], A.C. van Bochove, and P.T. de Boer

[1] University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands
r.o.taniman@ewi.utwente.nl

Nowadays, multicarrier transmission technique, e.g., optical OFDM, receives quite intense attention in the optical communications and networking field due to its potential to provide dispersion compensation. Adaptive modulation level is commonly and naturally employed in relation to multicarrier transmission, provided (sub)channel state information is available. The modulation level adaptation is usually done by a bitloading mechanism. In our work, given some modulation formats that are to be used, we attempt to reduce the computational complexity order of the bitloading algorithm from the one that has been presented in the literature. The bitloading algorithm is applied to a multicarrier multimode PON.

Introduction

Multicarrier transmission technique offers a potential to provide dispersion compensation from a rather different perspective where signal distortion is mitigated by way of parallel narrowband transmissions. This technique is beneficial for a multimode PON which has to offer a system capacity in the order of Gbps and cover few kilometers distances. Especially in the case of optical OFDM where some inherent digital signal processing is used, it's quite straightforward to use an adaptive modulation scheme: temporally adapting the modulation level of each subcarrier to optimize the system capacity, subject to a certain prescribed probability of bit error, based on the (sub)channel state information. In our multicarrier multimode PON, the modulation level adaptation, as a part of the dynamic capacity allocation [1], is determined by the OLT where it is assumed that the (sub)channel state information is available to the OLT.

Choosing the modulation level of each subcarrier is usually called bitloading. In [2], an optimal algorithmic bitloading was presented. The complexity of this bitloading, assuming the use of linear search, is $O\left(N^2K\right)$, where N is the total number of subcarriers and K is the maximum allowable number of bits per symbol. The bitloading is usually used as a part of a heuristic dynamic capacity allocation algorithm. In this case, a low-order algorithmic complexity is required as the algorithm is to be used in a real-time system (e.g., adaptive to (sub)channel condition and/or traffic load). Indeed some paper authors, e.g., [3, 4] became reluctant to employ an adaptive optimal bitloading scheme due to its algorithmic complexity, hence preventing them to further optimize the system capacity. In this paper, we describe our attempt to reduce the complexity of the bitloading algorithm, especially as a part of our dynamic capacity allocation algorithm as described in [1].

Adapting the bitloading algorithm

The optimal bitloading algorithm (**`Algorithm MinCost`**) given in [2] (see Algorithm 1) is of the cost minimization type (given a certain total number of bits b_{fixed} to be loaded). The **`Algorithm MinCost`** in fact works by using a greedy approach, i.e. loading b_{fixed} bits one by one, based on the least additional cost $\Delta c_{j,k} \equiv c_{j,k+1} - c_{j,k}$ at each step ($c_{j,k}$ is the cost to load k bits onto subcarrier j and $c_{j,0} \equiv 0\ \forall j$, meaning $c_{j,K} = \sum_{k=0}^{K-1} \Delta c_{j,k}$). Note that **`Algorithm MinCost`** requires $\Delta c_{j,k}$ be strictly increasing in k. For our multicarrier multimode PON we instead would like to maximize the number of loaded bits while not exceeding a certain cost c_{fixed} (**`Problem MaxBit`**). Nevertheless, it is possible to employ **`Algorithm MinCost`** in order to find a solution to **`Problem MaxBit`**, given c_{fixed}, by increasing the b_{fixed} one by one while the resulting $c^* \leq c_{\text{fixed}}$ until the state after which increasing the b_{fixed} one bit more would mean that the resulting $c^* > c_{\text{fixed}}$. We then have $b^* = b_{\text{fixed}}$ of this state.

Algorithm 1 **`Algorithm MinCost`**

Require: $\Delta c_j(b_j) = [c_j(b_j+1) - c_j(b_j)]$; $c = 0$ and for all j, let $b_j = 0$

while $\sum_j b_j < b_{\text{fixed}}$ **do**
  $j^* \leftarrow \arg\min_j \Delta c_j(b_j)$
  $c \leftarrow c + \Delta c_{j^*}(b_{j^*})$
  $b_{j^*} \leftarrow b_{j^*} + 1$
end while

Is the solution found using (repeated applications of) the **`Algorithm MinCost`** for the **`Problem MaxBit`** optimal? We establish the optimality of this solution as follows: Suppose there is another **`Algorithm alt`** that can load $b_{\text{alt}} > b^*$ with $c_{b_{\text{alt}}} \leq c_{\text{fixed}}$. Using **`Algorithm MinCost`** with b_{alt} bits to be loaded would result in $c^*_{b_{\text{alt}}} > c_{\text{fixed}}$. However, the optimality of **`Algorithm MinCost`** dictates that given $b_{\text{fixed}} = b_{\text{alt}}$, no other algorithms can satisfy / result in $c < c^*_{b_{\text{alt}}}$, meaning, $c \geq c^*_{b_{\text{alt}}}$ must hold which then implies that $c > c_{\text{fixed}}$. Because **`Algorithm alt`** is among these other algorithms, it implies that $c_{b_{\text{alt}}} > c_{\text{fixed}}$ too: a contradiction. Hence, it can be established that repeated applications of **`Algorithm MinCost`** results in an optimal solution to **`Problem MaxBit`**.

Algorithm 2 **`Algorithm MaxBit`**

Require: $\Delta c_j(b_j) = [c_j(b_j+1) - c_j(b_j)]$; $c = 0$ and for all j, let $b_j = 0$

while true **do**
  $j^* \leftarrow \arg\min_j \Delta c_j(b_j)$
  $c \leftarrow c + \Delta c_{j^*}(b_{j^*})$
  if $c > c_{\text{fixed}}$ **then**
    break {finish the while loop}
  else
    $b_{j^*} \leftarrow b_{j^*} + 1$
  end if
end while

Exploiting the greedy approach of **`Algorithm MinCost`**, modifying **`Algorithm MinCost`** in order to find an optimal solution to **`Problem MaxBit`** can be quite straightforward, namely, by testing whether ($c^*_b \leq c_{\text{fixed}}$ and) $c^*_{b+1} > c_{\text{fixed}}$ for $b = 0, 1, 2, \ldots$ increasingly (where c^*_b is the minimum cost to load b bits). The value of b that makes the test return true is then the value of b^*, i.e. the maximum total number

of bits that can be loaded given a fixed cost of c_{fixed}. We call the resulting algorithm: **`Algorithm MaxBit`** (see Algorithm 2).

Reducing the complexity order of the bitloading algorithm

Due to line 7 (loading additional bit) in Algorithm 2, the (linear) search for the minimum additional cost at line 2 must be repeated every time a bit is loaded. This repetition renders the complexity order rather high, namely, $O\left(N^2K\right)$. in order to reduce the complexity order, we propose **`Algorithm MaxBit2`** (see Algorithm 3) which employs a sorting in ascending order of the additional costs $\Delta c_{j,k}$ that needs to be performed only once. To enable the sorting, the $\Delta c_{j,k}$ should be precalculated for all j and k. Subsequent necessary operations for the bitloading are given at line 2-10.

Algorithm 3 **`Algorithm MaxBit2`**

Require: arrays A_j (containing data pairs: `subcarrier` and `cost`); $i = 1$; $c = 0$ and for all j, let $b_j = 0$

```
 1: construct the sorted array A using cost fields as the keys
 2: while true do
 3:     j* ← A[i].subcarrier
 4:     c ← c + A[i].cost
 5:     if c > c_fixed then
 6:         break {finish the while loop}
 7:     else
 8:         b_j* ← b_j* + 1
 9:         i ← i + 1
10:     end if
11: end while
```

`Algorithm MaxBit2` for the bitloading works correctly if $\Delta c_{j,k}$ is strictly increasing with k. This requirement comes from **`Algorithm MinCost`** as the basis for **`Algorithm MaxBit2`**. By using the modulation formats in Table 1 for the adaptive modulation scheme and following the common practice to use square signal constellations, one can satisfy this requirement (note that the bits are consequently loaded per 2 bits and $\Delta c_{j,k} \equiv c_{j,k+2} - c_{j,k}$ with k is even). From the table, $\hat{a}_k$ is the signal amplitude of an outermost point in the signal constellations which is proportional to the average energy of this constellation point. The last row of the table is a trick to ensure that no more than 8 bits can be loaded onto a subcarrier. For our specific system, we set $c_{j,k} = g_j \cdot \hat{a}_k$, where g_j is the overall gain of subchannel j, in order to avoid clipping.

TABLE 1: The modulation formats and their associated $\hat{a}_k$.

k bits/symbol	modulation format	$\hat{a}_k$
0	-	0
2	QPSK	1
4	16-QAM	3
6	64-QAM	7
8	256-QAM	15
10	-	∞

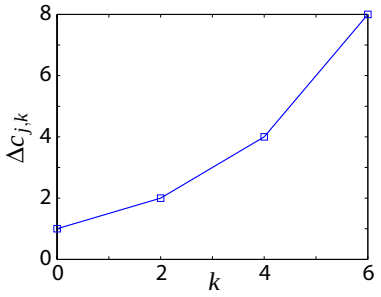

FIGURE 1: $\Delta c_{j,k}$ with $g_j = 1$ and $K = 8$

Because array A consists of NK elements, using an ordinary efficient sorting results in $O(NK\lg(NK))$ complexity order for **`Algorithm MaxBit2`**. This is already an improvement from $O\left(N^2K\right)$ (when N is large enough and K is small - this is a typical

system configuration). But by exploiting the fact that $\Delta c_{j,k}$ is strictly increasing with k, we can further reduce the resulting complexity order. First, we put $\Delta c_{j,k}$ into subarrays A_j for each j. Then, by using the merging process of the well-known merge sort (see e.g., [5]), we construct the sorted array A. The trick is to start the merging process from arrays A_j rather than from individual elements because each A_j is already naturally sorted. As the merging process can be illustrated as a binary tree (see Figure 2 as an example), the number of merging stages can be found to be $\lceil \lg(N) \rceil$. In each stage, at most NK comparisons need to be done. Now, it becomes clear that the resulting complexity order of this peculiar sorting is $O(NK \lg(N))$ which becomes the complexity order of **`Algorithm MaxBit2`** as it is dominated by the complexity order of the sorting part (at line 1).

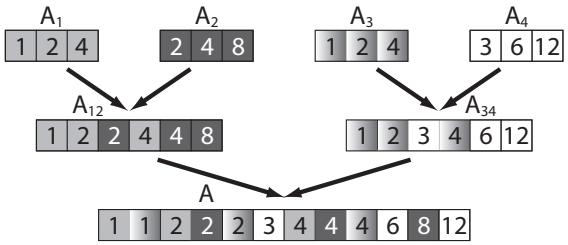

FIGURE 2: A sorting example with $N = 4$ and $K = 3$

Conclusion

We have shown that through some modifications, we were able to reduce the computational complexity order of the bitloading algorithm from $O\left(N^2K\right)$ as listed in the literature to $O(NK \lg(N))$. This complexity reduction can lend support to the use of the bitloading algorithm to further increase the system capacity of a multicarrier multimode PON.

Acknowledgements

This work is supported by the Dutch Ministry of Economic Affairs in the IOP GenCom project "Full Service Access Networks using Multimode Fibre". The authors would also like to thank Dr. Q. Yu for very useful discussions.

References

[1] R.O. Taniman, A.C. van Bochove, P.T. de Boer and B. Sikkes, "Dynamic capacity allocation for low-cost multicarrier multimode PON", in *Proc. 33rd European Conf. on Optical Communication*, Berlin, Germany, Sep. 2007, vol. 2, pp. 55-56.

[2] C.Y. Wong, R.S. Cheng, K.B. Letaief and R. Murch, "Multiuser OFDM with adaptive subcarrier, bit and power allocation," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 10, pp. 1747-1758, Oct. 1999.

[3] D. Kivanc, G. Li and H. Liu. "Computationally efficient bandwidth allocation and power control for OFDMA," *IEEE Trans. Wireless Communications*, vol. 2, no. 6, pp. 1150-1158, Nov. 2003.

[4] J. Gross, H. Geerdes, H. Karl and A. Wolisz, "Performance Analysis of Dynamic OFDMA Systems with Inband Signaling", *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 427-436, Mar. 2006.

[5] D.E. Knuth, The Art of Computer Programming vol. 3, 2nd ed., Reading, Massachusetts: Addison-Wesley, 1998, ch. 5, pp. 158-168.