Treemap-based Burst Mapping Algorithm for Downlink Mobile WiMAX Systems

Vanderpypen, Joël; Schumacher, Laurent

Published in:
Proceedings of the IEEE 74th Vehicular Technology Conference (VTC2011-Fall)

Publication date:
2011

Document Version
Early version, also known as pre-print

Treemap-based Burst Mapping Algorithm for Downlink Mobile WiMAX Systems

Joël Vanderpypen, Laurent Schumacher
FUNDP - The University of Namur, Belgium
Computer Science Faculty
{jva, lsc} @ info.fundp.ac.be

***Abstract*—This paper presents our sqTM burst mapping algorithm for downlink Mobile WiMAX systems. Based on a treemap visualization algorithm, we introduce a new burst mapping scheme we called sqTM, greatly reducing the amount of wasted slots. We obtained between 60% and 75% reduction compared to some reference algorithms we implemented. These limited wastes are able to provide better cell throughput and larger cell capacities. Unfortunately, sqTM is significantly slower than reference algorithms, but still easily coping with 5 ms frames.**

I. Introduction

Mobile WiMAX [1] has been pushed forwards by the IEEE as the standard for Broadband Wireless Access. It is based on Orthogonal Frequency Division Multiple Access (OFDMA), so the frequency band is divided into many subcarriers. Considering Partial Usage of the SubChannels (PUSC), which is the most common mode [2], the subcarriers are logically permuted before being gathered into subchannels. As a result, intercell interferences and the effect of fast fading are reduced.

The radio resource is the frame, which represent the usage of several subchannels for a 5 ms duration. A frame is divided into two parts: the downlink (DL) and the uplink (UL) subframes. In Time Division Duplexing (TDD), DL and UL subframes are time alternated, while in Frequency Division Duplexing (FDD), they use different frequency bands.

Resource allocation is performed by slots, which represent the usage of one subchannel for one symbol duration. Depending of the channel quality, a given user will use a specific PHY-profile, which determines the throughput he/she benefits from a slot. The allocation is done per burst, where a burst contains the data of one or multiple users, as long as a single PHY-mode is used per burst.

The burst mapping problem is quite different in DL and UL. After a one symbol duration preamble, the DL subframe is composed of a Frame Control Header (FCH) and the DL- and UL-MAPs, which are maps specifying the burst profile of each allocation. Then each burst receives a rectangle allocation, whose shape and position are specified through the Information Elements (IEs) of the DL-MAP. The more there are bursts, and the bigger the DL-MAP is. So it is preferable to gather users into a minimal number of bursts, the unique connection identifier (CID) separating them. In UL, the bursts are simply mapped as contiguous slots in a row wise order.

J. Vanderpypen acknowledges the funding support of the Fonds à la formation pour la Recherche dans l'Industrie et dans l'Agriculture (F.R.I.A.)

Since the UL mapping problem is straightforward, we focus here on DL. This tiling problem is actually a bi-dimensional bin packing problem, a variation of the knapsack problem, which has been proven NP-Hard [3]. The authors of [4] presented a heuristic where bursts size are rounded to a multiple of the height of the frame, leading to a consequent waste of slots. There are other strategies like SDRA [5] or the Raster-based algorithm [6], with virtually no wasted slots, but they multiply the number of bursts, and therefore increase significantly the DL-MAP signaling overhead. The authors of [7], [8] came up with some variations of what they called the Sequential Rectangle Placement (SRP). They define three sizes of bursts, and gather them into vertical stripes of growing width, according to their size category. The OCSA algorithm [9], [10] introduces an other stripe gathering strategy, more complex, but leading to fewer wasted slots.

In this paper, we consider a treemap visualization algorithm [11], and tune it to meet the WiMAX requirements to propose a more efficient heuristic named *sqTM*. Like OCSA, we gather bursts into vertical stripes, but of growing width, trying to limit the amount of wasted slots. As a result, our algorithm is able to seriously reduce the waste of slots while not increasing the DL-MAP. The rest of this paper is organized as follows. Treemap visualization is first introduced in Section II, then our sqTM algorithm is explained in Section III. Section IV details the reference algorithms we implemented into MATLAB scripts. Afterwards some computer simulation results are presented in Section V, first to compare algorithms on a single snapshot allocation, then to present averaged results. Finally some conclusions are drawn in Section VI.

II. Treemap visualization

Treemap visualization has been initially designed to visualize hard disk usage, in order to ease the identification of large files. The tree structure of a file system is represented by rectangles fitted into each other, whose area is proportional to the disk usage of the directory they represent.

A simple algorithm presented in [12] lists the size of all directories at the root level and divides the original rectangle representing the whole disk into different slices, whose width is proportional to the size of all first level directories. A special slice to represent free space has to be added. And then, recursively, for each directory, its area is cut into smaller slices representing all its sub-directories. The cuts should be done

alternatively horizontally and vertically. One drawback of this visualization algorithm is that it can lead to thin rectangles, whereas one would prefer rectangles with aspect ratio close to one (nearly square rectangles), for a better visualization.

The authors of [11] present a Squared Treemap algorithm, dividing a rectangle into smaller nearly squared rectangles, by cutting both horizontally and vertically. The idea is to cut a first vertical stripe into the original rectangle, whose area represents the first directory. Then an other directory is added to that stripe, which therefore becomes wider. The different directories of a stripe are divided by horizontal cuts. And so on, other directories are added to that stripe, until their aspect ratio can not be improved anymore. Then other vertical stripes are cut and filled with the remaining directories. This algorithm is a heuristic of linear time complexity.

III. Our sqTM algorithm

Even if we are not interested by the aspect ratio of the bursts, the procedure of the Squared Treemap algorithm seemed us pretty useful for the burst mapping problem. However, its application is not straightforward. Indeed, different bursts can not share the same subchannel during the same symbol time. So ceiling operations have to be considered, both at subcarrier scale on frequency dimension and at symbol scale on time dimension. These ceiling operations lead to wasted slots, whose quantity has to be minimized. Our allocation criteria is therefore not the aspect ratio, but the number of wasted slots.

A. The allocation procedure

Bursts are treated sequentially, and do not need to be sorted out. The allocation is done through vertical stripes of variable width. Each stripe is first composed of a single burst, with full height and rounded width. The number of wasted slots is evaluated. Then we sequentially try to add each of the unallocated bursts. If adding the burst reduces the number of wasted slots, it is added to the stripe. When adding any of the remaining bursts can not reduce the number of wasted slots, we start a new stripe, and try each unallocated burst.

The first stripe is built to include the FCH and the DL-MAP.

B. Ceiling operation

This section details how the resources are allocated to a stripe, and shared to its different bursts. Let us consider a stripe R composed of bursts b_i with $i = 1, ..., n_B$, each burst requiring r_i slots. The height of the frame is denoted by H.

Firstly, an accurate allocation is computed, with no ceiling, as Fig. 1(a) shows. The temporary width of the stripe $\tilde{w}_R$ is obtained as follows:

$$\tilde{w}_R = \frac{\sum_{b_i \in R} r_i}{H} \tag{1}$$

and the temporary height $\tilde{h}_i$ of each burst is given by

$$\tilde{h}_i = \frac{r_i}{\tilde{w}_R} \quad \forall i = 1, \cdots, n_B. \tag{2}$$

Secondly, the height allocated to each burst has to be rounded to h_i, to match subchannel scale, paying attention that the sum of all rounded heights must be equal to the height of the frame. Defining the operator $\lceil \cdot \rfloor$ such as it rounds to the closest integer, we have:

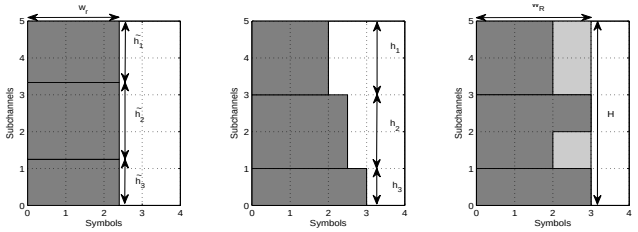
$$h_i = \begin{cases} \left\lceil \tilde{h}_i \right\rfloor & \forall i = 1, ..., n_B - 1; \\ H - \sum_{j=1}^{n_B - 1} h_j & i = n_B. \end{cases} \tag{3}$$

As shown on Fig. 1(b), the width of each burst has to be adapted consequently to $\tilde{\tilde{w}}_i$:

$$\tilde{\tilde{w}}_i = \frac{r_i}{h_i} \quad \forall i = 1, \cdots, n_B. \tag{4}$$

And thirdly, the width of the stripe w_R, which is also the final width w_i of each burst, can be fixed to the closest larger integer:

$$w_R = \max_{i=1,\cdots,n_B} \left\lceil \tilde{\tilde{w}}_i \right\rceil. \tag{5}$$



(a) Accurate allocation (b) Rounded heights (c) Ceilled allocation

Fig. 1. Ceiling operation, with bursts of size 4, 5 and 3 slots. After ceiling, 3 slots are wasted.

The number t_{wasted} of slots wasted in the stripe allocation of Fig. 1(c) can be estimated by

$$t_{wasted} = (H \cdot w_R) - \sum_{i=1}^{n_B} r_i. \tag{6}$$

To limit computations, we can perform a single test before the full ceiling operation, which is only fruitful if adding the new burst reduces the number of wasted slots. After computing (1), even if rounded heights shown on Fig. 1(b) would not lead to any waste, we loose at least $H \cdot (\lceil \tilde{w}_R \rceil - \tilde{w}_R)$ slots. This is a minimum bound on the amount of wasted slots for the stripe. If this bound is greater than the amount of wasted slots of a previous stripe composition, other ceiling computations are useless, we know the current stripe composition does not produce better results than the previous one.

C. Algorithm

We therefore came up with the Algorithm 1. While there are still unallocated bursts, it makes a new stripe, sequentially trying to add all the unallocated bursts. If the stripe is empty, the burst is directly added to the stripe (lines 4 to 6). If not, the ceiling of the previous section has to be done. The test of line 9 prevents the algorithm to perform some computations (lines 10 to 12, plus the evaluation of line 13) when we know they are pointless. Finally, after all computations, if wasted slots are reduced by adding burst b_i (line13), it is definitely added to the stripe, and other bursts are then tried to be added.

Algorithm 1 sqTM

while some unallocated bursts remain **do**
  Start a new stripe R;
  for each unallocated burst b_i **do**
    if stripe R is empty **then**
      Add b_i into R;
      $t_{wasted} \leftarrow (H - r_i \bmod H)$
    else {R is non empty}
      $\tilde{w}_R \leftarrow \sum_{b_j \in R \cup \{b_i\}} r_j / H$;
      if $(\lceil \tilde{w}_R \rceil - \tilde{w}_R) \cdot H < t_{wasted}$ **then**
        $h_j \leftarrow \lceil r_j / \tilde{w}_R \rfloor \quad \forall b_j \in R$;
        $h_i \leftarrow H - \sum_{b_j \in R} h_j$;
        $w_R \leftarrow \max_{b_j \in R \cup \{b_i\}} \lceil r_j / w_R \rceil$;
        if $(H \cdot w_R) - \sum_{b_j \in R \cup \{b_i\}} r_j \leq t_{wasted}$ **then**
          Add b_i into R;
          $t_{wasted} \leftarrow (H \cdot w_R) - \sum_{b_j \in R \cup \{b_i\}} r_j$;
        end if
      end if
    end if
  end for
end while

D. Complexity

The algorithm is composed of a main while loop, limited by the number n of bursts. Then for each stripe, all unallocated bursts are tested, which number is also bounded by n. As a result, the complexity of our sqTM algorithm is $\mathcal{O}(n^2)$.

IV. Some reference algorithms

To evaluate the performance of sqTM, we have implemented four other burst mapping schemes, namely the bucket-based algorithm [4], SDRA [5], SRP [7] and OCSA [9].

A. Bucket-based

The authors of [4] proposed a simple algorithm, where burst sizes are ceiled to a multiple of H, the number of subchannels, and are handled sequentially. Bursts receive as many one-symbol width columns as they require. This algorithm is of complexity $\mathcal{O}(n)$. In average, it wastes $\frac{1+H}{2}$ slots per burst since the last column of each burst can contain 1 to H slots.

B. SDRA

The SDRA algorithm of [5] has the same column filling strategy as the bucket-based algorithm, but here when the allocation of one burst is done, it continues filling the same one-symbol width column with the allocation of the next burst. As a result, no slots are wasted by any ceiling operation, and this algorithm is also of complexity $\mathcal{O}(n)$.

However, to meet the rectangle shape allocation, each burst actually receive multiple rectangles. Depending on its size, each burst can receive up to 3 rectangles: one to complete the previous column, one with several full columns, and one for the end of the allocation on a new column. Because the size of the DL-MAP depends on the number of rectangle allocations, even if this algorithm does not waste any slot by ceiling operations, many slots are wasted on DL-MAP signaling.

C. SRP

The authors of [7] modeled the resource allocation problem as a Sequential Rectangle Placement. They defined three classes of bursts, with W standing for the number of symbols of the downlink scheduling period:

- the small bursts, with $r_i \leq 2\sqrt{H}$. These ones can only have one-symbol width allocation.
- the medium bursts, with $2\sqrt{H} < r_i \leq \frac{W}{2}\sqrt{H}$, receiving maximum $\frac{W}{2}$-symbol width allocation.
- the large bursts, with $r_i > \frac{W}{2}\sqrt{H}$, with maximum $2W$-symbol width allocation.

They also defined the concept of job sets, which are column shaped and contain only bursts of the same kind. The maximum width of a job set R is the maximum width of its bursts, and is denoted by $\mathrm{MAX}(R)$. The width of a job set R is denoted $\mathrm{WIDTH}(R)$ and is evaluated by:

$$\mathrm{WIDTH}(R) = \min \left\{ \left\lceil \frac{\sum_{b_j \in R} r_j}{H - |R|} \right\rceil \ , \ \mathrm{MAX}(R) \right\} \quad (7)$$

They sequentially handle each burst, trying to add it within the current job set of its size. If there remains not enough room, they close the job set, and start a new one. So SRP complexity is $\mathcal{O}(n)$, even if it requires more computations than the two previous algorithms.

D. OCSA

The authors of [9], [10] presented an algorithm where bursts have to be sorted in a descending order (largest area first). The largest unallocated burst is placed first, in a vertical stripe as narrow as possible. Then on the top of that burst, the algorithm tries to place the maximum amount of bursts fitting in the stripe without modifying its width. When no more burst can be added on the top of it, an other stripe is started with the largest unallocated burst remaining, and so on.

Since there can be n stripes, and for each one the n bursts can be tested, the total complexity of OCSA is $\mathcal{O}(n^2)$.

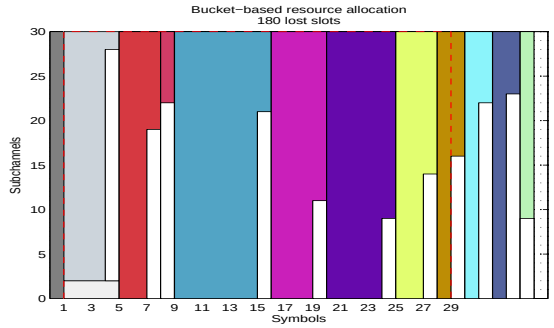
V. Performance evaluation

We implemented both our sqTM algorithm and the four reference ones into MATLAB scripts. We will detail first the parameters of these simulations, then we will present some allocation results. Afterwards, averaged results on the amount of wasted slots and computational time will be presented.

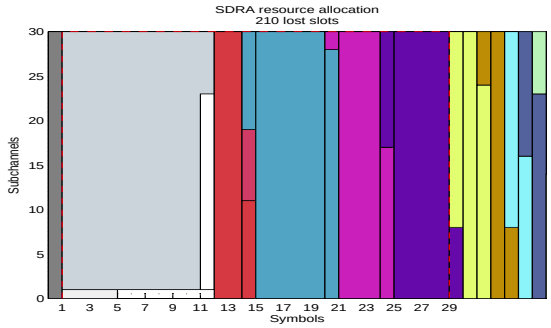
A. Simulation parameters

We focused here on a TDD WiMAX system, with an DL:UL ratio of 2:1. So the DL subframe lasts a 29-symbol duration, whose first one is preamble. We considered a 10 MHz bandwidth divided into 30 subchannels. As a result, the surface we have for burst mapping is of size 28×30, giving 840 slots.

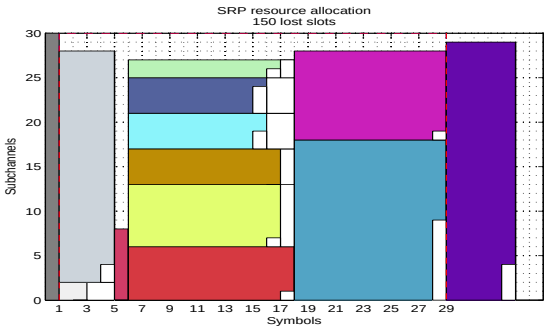
As [13] mentions, the FCH represents 24 bits. The DL-MAP has a fixed 88-bit part, plus 60 extra bits for each burst. The FCH and the DL-MAP are repeated 4 times and are transmitted with QSPK 1/2. Since we do not consider any UL subframe, only a burst of random size will act as the UL-MAP.



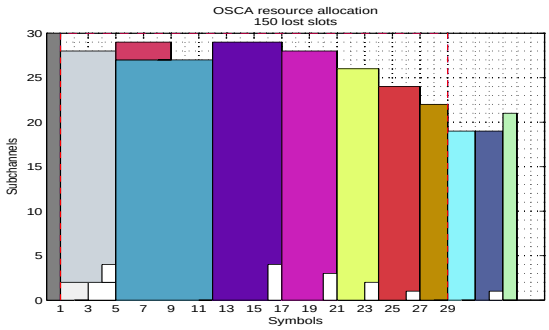
(a) Bucket-based



(b) SDRA



(c) SRP



(d) OCSA

Fig. 2. Reference algorithms sample allocation

We considered the 9 Modulation and Coding Scheme (MCS) levels of [14], from QPSK 1/2 to 64QAM. To reduce the DL-MAP size, we gathered all users of the same MCS into the same burst. As a result, there are the FCH, the DL-MAP, 9 DL bursts plus an extra burst for the UL-MAP to place on the DL-subframe. With the 0.937 bit/s/Hz spectral efficiency of QPSK 1/2 [13], FCH requires 4 slots, and a 10-burst DL-MAP requires 102 slots.

In this primary work, we wanted to focus of burst mapping efficiency, and compare our new technique with some references, without interference from different scheduling techniques. For that purpose, the size of each burst has been generated randomly, such that the sum of each burst requirement plus the FCH and a 10-burst DL-MAP correspond to the 840 available slots. As a consequence, due to ceiling operations or increased DL-MAP size due to SDRA's divided bursts, all bursts can not fit on the subframe. We will compare the number of overflowing slots for each algorithm.

B. A sample output result

First let us present a single snapshot allocation. The slots have been randomly shared between bursts as Table I details. The allocation produced by reference algorithms are presented in Fig. 2, while Fig. 3 shows our sqTM allocation. The first dark grey stripe is the preamble, and the two light grey rectangles of the lower left corner are the FCH and the DL-MAP. Reference algorithms waste a lot more slots. The bucket-based algorithm, shown in Fig. 2(a), wastes 180 slots, with in average $\frac{1+H}{2}$ wasted slots per burst. SDRA achieves to virtually waste no slot at all, but because it splits the 10 bursts into 23 rectangle allocations, the DL-MAP is more than doubled, and as Fig. 2(b) shows, 210 slots overflow from the burst allocation area. Both SRP and OCSA manage to limit the waste to 150 slots in a quite similar way. With its strategy of growing width stripes, SRP makes large stripes of multiple bursts (Fig. 2(c)), while OCSA and its narrowest stripe strategy make a lot of thin stripes (Fig. 2(d)).

FCH	DL MAP	UL-MAP	Burst #1	Burst #2	Burst #3
4	102	71	8	189	109
Burst #4	Burst #5	Burst #6	Burst #7	Burst #8	Burst #9
141	76	44	38	37	21

TABLE I
SAMPLE SNAPSHOT OF BURST SIZES IN SLOTS.

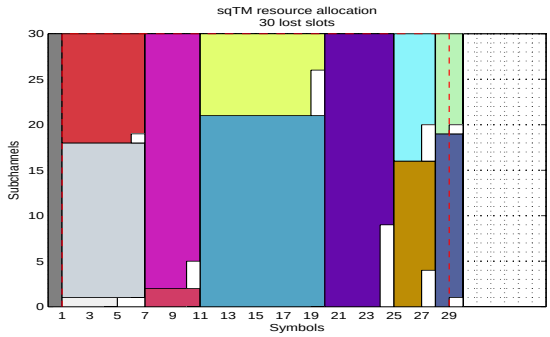


Fig. 3. sqTM sample allocation

As one can see on Fig. 3, sqTM manages to ceil bursts needs with a very limited waste. For this snapshot, all the unused slots of the burst allocation area correspond to only one overflowing column of 30 slots wasted by sqTM. This performance can be achieved thanks to our growing stripe strategy, where decisions are taken to minimize wasted slots.

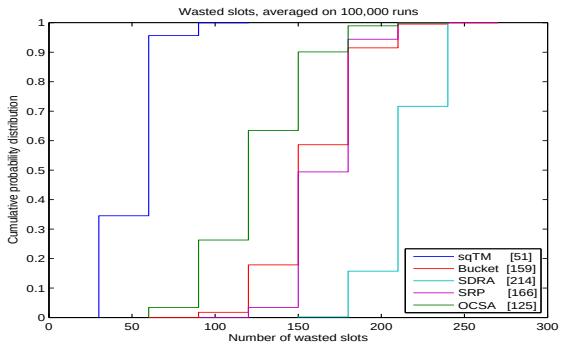


Fig. 4. Wasted slots distributions.

C. Averaged results

We also averaged the number of wasted slots by each algorithm on $100,000$ runs. The cumulative probability function and the mean of these wastes are presented in Fig. 4. Distributions are not smooth curves since wastes can only be a multiple of the number of subchannels (30 here). As one can see, sqTM manages to reduce the amount of wasted slots by 60% compared to OCSA, or even by 75% compared to SDRA.

Considering the 840 slots available for allocation, it means that in average, sqTM manages to only waste 6% of the slots, while the bucket-based algorithm, SDRA, SRP and OCSA respectively waste 19%, 25%, 20%, and 15% of the available slots, which seems quite impressive.

D. Complexity

We already demonstrated the important reduction of wastes sqTM can provide, but let us now have a look at its complexity. Even if it has a $\mathcal{O}(n^2)$ complexity like OCSA while the others three are in $\mathcal{O}(n)$, this is not a real issue since users can be gathered into a limited number of bursts, according to their channel quality. Actually, running time is more relevant. Table II presents the running time of our MATLAB scripts to perform 100,000 allocations. As one can see, OCSA in $\mathcal{O}(n^2)$ is as fast as SDRA which is in $\mathcal{O}(n)$.

As far as sqTM is concerned, it required 36.7s to perform the allocations, so it is 2 times slower than OCSA, or even 6 times slower than the bucket-based allocation. Computations are heavier and take more time with sqTM than with the other reference algorithms, but 36.7s for 100,000 allocations corresponds to $367\mu s$ per allocation. This is much shorter than the 5ms frame duration so it is not really an issue.

VI. Conclusions

In this paper, we tackled the problem of burst mapping in DL WiMAX systems, after scheduling been performed, when it comes to allocate specific subchannels for a few symbol durations. This problem is actually NP-Hard, and only a few heuristics have been proposed to solve it. Allocations must have a rectangular shape, and match both subchannel and symbol scales. Needs therefore have to be ceiled, and this leads to wastes which we have to limit.

We introduced a new burst mapping scheme named sqTM. It is based on a treemap visualization algorithm. From our numerical simulations, we obtained between 60% and 75% reduction of wasted slots compared to the reference algorithms we implemented. These limited wastes are able to provide better cell throughput and larger cell capacities from the same radio resources. As far as the running times are concerned, it appears our algorithm is significantly slower, but still fast enough to meet the time constraint of 5 ms frames.

For future work we should consider more deeply how to deal with these wasted slots and the overflow they cause. They are considered allocated by the scheduling algorithm, but actually can not be used. If bursts have to be dropped, it seems be preferable to drop the bursts of the lowest PHY-modes to maintain the highest cell throughput. The rescheduling of these bursts should be investigated.

Algorithm	Complexity	Running time
sqTM	$\mathcal{O}(n^2)$	36.7 s
Bucket	$\mathcal{O}(n)$	5.7 s
SDRA	$\mathcal{O}(n)$	22.8 s
SRP	$\mathcal{O}(n)$	27.9 s
OCSA	$\mathcal{O}(n^2)$	19.3 s

TABLE II
Running time for 100,000 allocations.

References

[1] "Draft standard for local and metropolitan area networks part 16: Air interface for broadband wireless access systems," IEEE Unapproved Draft Std P802.16 Rev2/D2, Tech. Rep., Dec. 2007.

[2] M. Maqbool, M. Coupechoux, and P. Godlewski, "Subcarrier Permutation Types in IEEE 802.16e," ENST (Télécom Paris), Tech. Rep., Apr. 2008.

[3] S. Martello and P. Toth, *Knapsack problems: Algorithms and computer implementations*. John Wiley & Sons, 1990.

[4] T. Ohseki, M. Morita, and T. Inoue, "Burst construction and packet mapping scheme for ofdma downlinks in ieee 802.16 systems," in *GLOBECOM '07*, 2007, pp. 4307 –4311.

[5] A. Erta, C. Cicconetti, and L. Lenzini, "A downlink data region allocation algorithm for ieee 802.16e ofdma," in *6th International Conf. on Information, Communications Signal Processing*, 2007, pp. 1–5.

[6] Y. Ben-Shimol, I. Kitroser, and Y. Dinitz, "Two-dimensional mapping for wireless ofdma systems," *Broadcasting, IEEE Transactions on*, vol. 52, no. 3, pp. 388–396, 2006.

[7] A. Israeli, D. Rawitz, and O. Sharon, "On the complexity of sequential rectangle placement in ieee 802.16/wimax systems," *Inf. Comput.*, vol. 206, pp. 1334–1345, November 2008.

[8] R. Cohen and L. Katzir, "Computational analysis and efficient algorithms for micro and macro ofdma scheduling," in *The 27th Conf. on Computer Comm. INFOCOM 2008.*, 2008, pp. 511–519.

[9] C. So-In, R. Jain, and A.-K. Al Tamimi, "Ocsa: An algorithm for burst mapping in ieee 802.16e mobile wimax networks," in *15th Asia-Pacific Conference on Communications APCC.*, 2009, pp. 52–58.

[10] C. So-In, R. Jain, A.-K. C. S.-I. Al Tamimi, R. Jain, A.-K. C. S.-I. Al Tamimi, R. Jain, and A.-K. Al Tamimi, "eocsa: An algorithm for burst mapping with strict qos requirements in ieee 802.16e mobile wimax networks," in *Wireless Days (WD), 2nd IFIP*, 2009, pp. 1–5.

[11] M. Bruls, K. Huizing, and J. J. van Wijk, "Squarified Treemaps," in *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization VisSym'99, Vienna (Austria)*, May 1999, pp. 33–42.

[12] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transactions on Graphics*, vol. 11, no. 1, pp. 92–99, Jan. 1992.

[13] C. So-In, R. Jain, and A.-K. Tamimi, "Capacity Evaluation for IEEE 802.16e mobile WiMAX," *J. Comp. Sys., Netw., and Comm.*, pp. 2:1–2:1, January 2010.

[14] T. Celcer, T. Javornik, and G. Kandus, "Fairness oriented scheduling algorithm with qos support for broadband mimo systems with heterogeneous traffic," Jozef Stefan Institute, Slovenia, COST 2100 TD(09)927, Sep. 2009.