Computational Analysis and Efficient Algorithms for Micro and Macro OFDMA Downlink Scheduling

Reuven Cohen Liran Katzir
Department of Computer Science
Technion–Israel Institute of Technology
Haifa 32000, Israel

***Abstract*— OFDMA is one of the most important modulation and access methods for the future mobile networks. Before transmitting a frame on the downlink, an OFDMA base station has to invoke an algorithm that determines which of the pending packets will be transmitted, what modulation should be used for each of them, and how to construct the complex OFDMA frame matrix as a collection of rectangles that fit into a single matrix with fixed dimensions. We propose efficient algorithms, with performance guarantee, that solve this intricate OFDMA scheduling problem by breaking it down into two sub-problems, referred to as macro and micro scheduling. We analyze the computational complexity of these sub-problems and develop efficient algorithms for solving them.**

I. Introduction

Orthogonal Frequency Division Multiple Access (OFDMA) is one of the most important modulation and multiple access methods for the future mobile networks. It is an extension of OFDM, which is today the modulation of choice for non-mobile wireless access systems such as IEEE 802.11 (WiFi) and IEEE 802.16 (WiMax). OFDM divides a single frequency band into dozens of sub-carriers for parallel transmissions by the same user. This division increases the tolerance to noise and multipath effects, while enabling more efficient use of bandwidth allocation. OFDMA extends OFDM by dividing the original band into many subchannels, each comprising a group of orthogonal carriers. Various modulations and FEC (Forward Error Correction) techniques are used for each subchannel, in order to provide improved coverage and throughput.

Unlike OFDM, OFDMA has many intricate constraints on its frame structure. The structure of an OFDMA downlink frame is depicted in Figure 1. The frame can be viewed as a 2-dimensional matrix, with the y-axis indicating the number of subchannels, each consisting of several orthogonal and not necessarily adjacent frequencies, and the x-axis indicating the time. The frame starts with a Frame Control Header (FCH), which contains information about the code rate, modulation level, and the length of the downlink (DL) and uplink (UL) maps. The data messages (PDUs) are transmitted in multiple bursts. There are 7 bursts shown in Figure 1. Each burst is transmitted by the base station using a specific combination of modulation technique, code rate, and error correcting codes. Such a combination is referred to as PHY-profile. Due to OFDMA-related constraints, each data burst must be allocated a rectangular region within the frame. A burst may contain one or more PDUs destined for one or more receivers. To ensure correct decoding of the received signals, certain PHY-profile information about every burst is required. This information is included as overhead in the DL burst map. The DL and UL maps are transmitted using a pre-determined and robust PHY-profile.

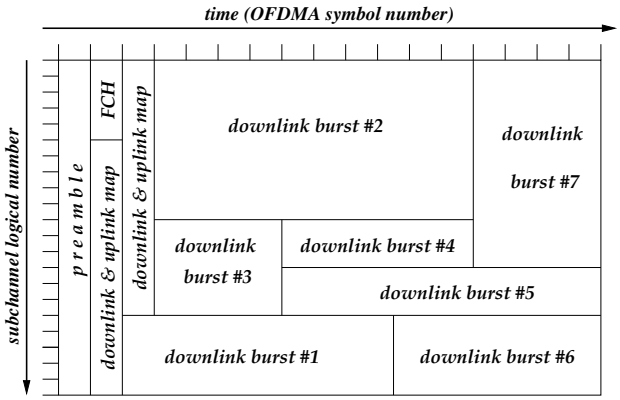


Fig. 1. The Structure of an OFDMA Downlink Frame

This paper studies combinational aspects of scheduling on an 802.16/WiMax downlink OFDMA channel. Before transmitting a downstream frame, typically once every few ms., the base station has to invoke a scheduling algorithm that will generate the frame matrix as shown in Figure 1. To this end, the scheduler needs to address the following decision problems:

- To decide which PHY-profile will be used for each PDU. There are several dozen potential profiles, each with its own bandwidth requirement and robustness against transmission errors. It is not possible to use all profiles in one frame. Moreover, each profile introduces a fixed significant overhead in the DL map field of the frame, and therefore, because of throughput considerations, the scheduler should try to minimize the number of profiles accommodated in every frame.
- To determine which PDU will be transmitted in the next frame. This decision has to take into account many factors, such as: (a) Quality of Service, since some of the PDUs have a guaranteed upper bound on their maximum delay; (b) total throughput maximization, since transmission to some of the hosts is difficult and requires more bandwidth for reliable delivery, and (c) fairness.
- To decide where exactly in the frame every burst will be located. Here there are also several constraints, some

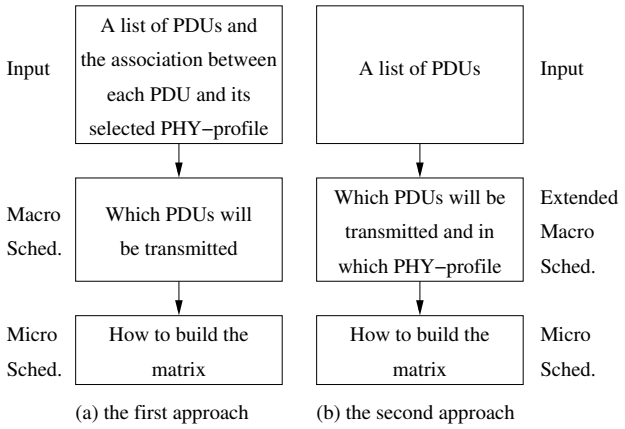


Fig. 2. The two approaches studied in this paper

of which are: (a) power boosting, namely the ability of the base station to increase the transmission power used for some burst while decreasing the power used for other bursts transmitted at the same time on different subchannels; (b) efficiency, since the requirement that every PHY-profile will be represented as a rectangle in the frame matrix may leave some unused space in each rectangle.

There are two kinds of difficulties associated with addressing these problems. The first is that a solution for each problem depends on the solutions for the other two. This leads to a circular dependency that should be broken. The second difficulty is that, as shown later, each of these decision problems is NP-hard, which means that even if the circular dependency is broken, finding an optimal scheduling algorithm is not possible under standard assumptions.

In this paper we address the OFDMA scheduling problem in two approaches. In the first approach (Section IV and Section V) we assume that the base station determines in advance the PHY-profile to be used for each PDU, and the profit (utility) gained from transmitting this PDU using its selected PHY-profile. Then, there are two phases. In the first phase, referred to as *macro scheduling*, the base station determines which of the PDUs will actually be selected for transmission. In the second phase, referred to as *micro scheduling*, the scheduler determines how to build the OFDMA matrix from the selected PDUs. In the second approach we extend the macro scheduler such that it also determines the PHY-profile for each PDU. This is referred to as *extended macro scheduling*. Figure 2 summarizes these two approaches.

The assignment of profit (utility) to PDUs, which is based on various considerations such as QoS, throughput maximization, fairness, and channel state information (CSI), is beyond the scope of this paper. We address this issue in [5] in the context of uplink transmission, and similar ideas can be implemented on the downlink as well. In what follows we give some examples for the profit assignment considerations discussed in [5]. A delay-sensitive (VoIP) packet is assigned a high profit when it is received by the scheduler. This priority is reduced during the time the packet is delayed before transmission. In contrast, data packets, which are not so much sensitive to delay, are assigned a lower profit, and their profit is only slightly affected by the time they wait for transmission. A packet whose intended channel is known to be good is assigned a higher profit than a similar packet whose intended channel is not as good. A packet protected by an ARQ (automatic repeat request) protocol is assigned a higher profit than a non-protected packet when the two are supposed to be transmitted over a bad channel.

The main contributions of this paper are:

- Breaking the OFDMA scheduling problem into two more tractable problems, referred to as macro scheduling and micro scheduling.
- Analyzing the computational complexity of the macro and micro scheduling problems and providing algorithms with the best performance guarantee for these problems.
- Developing efficient and practical algorithms for these NP-hard scheduling problems. Specifically, we provide an optimal macro scheduling problem, and an algorithm with only 2.5% overhead for the micro scheduling problem.
- Introducing a new concept in which the transmission profile of a PDU is not only determined by the channel condition of the intended receiver, but on the entire system conditions. To this end, we extend the macro scheduling problem to include the selection of transmission parameters for each PDU, and provide an efficient algorithm for this extended macro scheduling problem. This algorithm has the best possible approximation guarantee up to an arbitrary small constant. It provides a significant improvement over the algorithm for the macro scheduling problem, at the cost of higher running time.

The rest of this paper is organized as follows. In Section II we discuss related work. In Section III we divide the OFDMA scheduling problem into macro and micro scheduling sub-problems, and discuss their computational complexity. In Section IV we present efficient algorithms for the macro scheduling sub-problem and in Section V we present such algorithms for the micro scheduling sub-problem. Section VI presents an extension to the macro scheduling problem, which allows the base station to select a PHY-profile for every PDU as a part of the scheduling process. This extension is shown to significantly improve the performance of the scheduler. Section VII presents an approach for achieving global optimization using the previous algorithms. Section VIII presents a simulation study and Section IX concludes the paper.

II. Related work

While much work has been done on scheduling in wireless networks, only a few papers address resource allocation in OFDMA channels. In [2], some variations of the micro scheduling problem are addressed. In these variations, the input consists of bursts and their sizes. The bursts have priorities and the objective is to find an efficient allocation with accordance to these priorities. The main differences between [2] and our paper are that: (a) we analyze the computational complexity of the micro scheduling problem and propose also approximation algorithms while [2] presents only heuristics; (b) we address not only the micro scheduling problem but also the macro scheduling problem and the combination between the micro scheduling problem and the macro scheduling problem; and (c) we consider a more general case, where profit is assigned to each PDU and not to each burst (a collection of PDUs).

In [12], a variation of the micro scheduling problem is addressed. In this variation, the bursts have a specific order

in which they can be scheduled. That is, in order to schedule burst i, all bursts $1 \ldots i-1$ must also be scheduled. The authors provide both theoretical hardness results, and an approximation algorithm. The main differences between [12] and our paper are that: (a) we address not only the micro scheduling problem but also the macro scheduling problem and the combination between the two; and (b) in our paper the bursts are not scheduled based a predetermined order, but according to the profit of their PDUs.

In [18], it is shown that when an earliest-deadline-first greedy algorithm is implemented over good channels of a centralized wireless network, like the one we consider, the number of packets lost due to deadline expiration is minimized. In [16], the authors provide a fluid fair queueing scheduler for a noisy channel. In [19], the authors present an efficient fair queuing approximation of a noisy channel using deficit round robin, which takes less time to process. In [1], a scheduling algorithm that uses an N-state Markov model to characterize the channel is presented. This algorithm supports adaptive modulation and coding (AMC), which are used to adjust the modulation and FEC to the forecasted channel state. The idea of assigning higher data rates to hosts with a better channel – to maximize throughput while ensuring acceptable bit-error rate (BER) – is used by [9], [17].

In the uplink channel of an OFDM network, multiple hosts can transmit simultaneously over different sub-carriers. Since the channel characteristics for different users may be independent, dynamic assignment of sub-carriers to hosts can significantly improve the throughput [21], [22]. However, this "water filling" approach for maximizing instantaneous throughput does not take into account the QoS requirements of these packets, and it is therefore unsuitable for traffic of delay sensitive applications (like voice over IP). The authors of [15] address this problem in the context of OFDMA, by allocating sub-carriers to hosts in a way that satisfies the rate requirements of each host, while using minimum power. In [20], utility-based cross-layer optimization problems are defined. In [23], the authors provide an algorithm for OFDMA power assignment.

III. Problem Formulation and Complexity Classification

Consider a set of PDUs with arbitrary sizes awaiting transmission. Each PDU can be transmitted using several different PHY-profiles, where each such a profile includes a modulation technique, code rate, and FEC. There are two possible timings for selecting a PHY-profile for each PDU. The first is to determine the PHY-profile for each PDU in advance (a priori), and only then to choose the PDUs for transmission. The second is to determine first the profit of scheduling each PDU using each of the possible PHY-profiles, and only then to let the scheduler pick, on the fly, at most one instance for each PDU. In other words, with the a priori approach the selection of PHY-profiles is not performed by the macro scheduler, but by another algorithm that is executed before the macro scheduler has to make its decisions. We assume that even if the selection of a PHY-profile is performed a priori, the base station has the same information about the physical layer that it has when this selection is performed on the fly. Hence, the advantage, to be shown later, of using the on the fly approach, stems from the coupling between the selection of a PHY-profile for each PDU and the other tasks of the macro-scheduler. We address the a priori PHY-profile selection in Section IV, and the on the fly selection in Section VI.

The scheduling space in every frame is a two dimensional matrix, of T time slots and C OFDMA logical channels. Due to physical layer considerations, the scheduler must divide this matrix into rectangles, where each rectangle is used for the transmission of one or more PDUs from the same PHY-profile [11]. Each of these rectangles is referred to as a *burst.* In the beginning of the frame, each rectangle has an overhead for describing its exact location within the frame and its PHY-profile attributes. The size H of this overhead is constant for each burst. In other words, there are several PDUs within a burst and the overhead is per burst and not per PDU. Note that in the variant of the problem considered in this paper, every user must decode the entire downlink frame in order to decide which PDUs to read. There is another variant, where the map indicates which burst should be decoded by which user. In the latter variant, in addition to the fixed overhead per burst, there is a fixed overhead per user in every burst. Due to lack of space, we do not explicitly address the latter variant. However, the algorithms we present can be adjusted to solve it as well.

The goal of the scheduler is to maximize the profit gained by the transmitted PDUs in each frame. This goal requires the scheduler to make two types of decisions for every downstream frame:

- **MaSP:** A Macro Scheduling Decision: Deciding which PHY-profiles will be accommodated in this frame, and which PDUs will be transmitted for every selected PHY-profile, assuming that the association between PDUs and their PHY-profiles has already been determined.
- **MiSP:** A Micro Scheduling Decision: Deciding how many rectangles (bursts) will be used for each PHY-profile, and where to locate each rectangle within the frame.

We later show that the macro scheduling problem (MaSP) is equivalent to the well-known NP-hard Multiple-Choice Knapsack Problem [13]. The micro scheduling problem (MiSP) is associated with a tradeoff that has a critical impact on the performance of the downstream channel. On one hand, it is important to minimize the number of bursts for each PHY-profile, because each such burst has a significant overhead header. On the other hand, with smaller rectangles it is easier to minimize the bandwidth that is allocated to a rectangle but not fully used.

The table in Figure 3 classifies the two problems with respect to their computational complexity. Both problems are NP-hard. However, we prove that they can be solved in polynomial time within $(1+\epsilon)$ from their optimum, for any $\epsilon > 0$. The selection of ϵ has, of course, a critical effect on the running time of the scheduler. The last row in the table is related to the extended MaSP problem, to be discussed in Section VI.

IV. The Macro Scheduling Problem (MaSP)

We start with a formal definition of MaSP.

Problem 1 (MaSP):

Instance: The frame's size $L = T \cdot C$, the overhead H for using a PHY-profile, and a set $P = \{p_1, p_2, \ldots, p_n\}$ of PDUs awaiting transmission. Each PDU p_i is associated with a PHY-profile $PHY(p_i)$, a non-negative profit $p(p_i)$, and a non-negative size $s(p_i)$.

Problem	NP-hard	Approximation guarantee	Running time
MaSP (Macro Scheduling Problem)	Yes	$(1+\epsilon)$-approximation for every fixed ϵ	polynomial in $1/\epsilon$
MiSP (Micro Scheduling Problem)	Yes	$(1+\epsilon)$-approximation for every fixed ϵ	not polynomial in $1/\epsilon$
E-MaSP (Extended MaSP)	Yes	$(\frac{e}{e-1}+\epsilon)$-approximation for every fixed ϵ	polynomial in $1/\epsilon$

Fig. 3. Computational Complexity and Running Time for the Various Scheduling Problems

Objective: Find a feasible subset $P^{'} \subseteq P$ of PDUs with a maximum profit. The size of a subset P' is the cumulative size of its PDUs plus a penalty of H for every used PHY-profiles. A feasible subset is a subset whose size is $\leq L$. The profit of a subset P' is the cumulative profit gained by its PDUs.

We note that the well-known Knapsack problem is a special case of MaSP, with only one PHY-profile. The Knapsack problem is defined as follows:

Instance: A set S of items $s_1, s_2, \ldots, s_m$ and a capacity c. Each item s_i has profit $p(s_i)$ and a weight $w(s_i)$.

Objective: Find a subset $S^{'} \subseteq S$ of items such that this subset has a feasible packing, namely, $\sum_{s_j \in S'} w(s_j) \leq c$, and the aggregated profit $\sum_{s_j \in S'} p(s_j)$ is maximized.

We now show that MaSP is polynomially equivalent to the Multiple-Choice Knapsack Problem (MCKP). Therefore, an algorithm for MCKP can also solve MaSP with the same performance guarantee. MCKP is defined as follows:

Instance: Disjoint classes $N_1, \ldots, N_m$ of items, and a capacity c. Each item $s \in N_j$ has a profit $p(s)$ and a weight $w(s)$.

Objective: Find a subset S' of items *with at most one item from each class* that has a feasible packing, such that the aggregated profit $\sum_{s_j \in S'} p(s_j)$ is maximized.

Definitions:

1) An r-approximate solution for a maximization problem is a feasible solution whose profit is at least $\frac{1}{r}$ the profit of the optimal solution.
2) An r-approximation algorithm is an algorithm that always finds an r-approximate solution.
3) An approximation scheme is an algorithm that receives a parameter $\epsilon > 0$ along with the problem instance and produces a $(1+\epsilon)$-approximate solution for the specified instance.
4) A Polynomial Time Approximation Scheme (PTAS) is an approximation scheme whose running time is polynomial in the size of the input.
5) A Fully Polynomial Time Approximation Scheme (FPTAS) is a PTAS whose running time is also polynomial in $\frac{1}{\epsilon}$.
6) A pseudo-polynomial time algorithm is an algorithm whose running time is polynomial in the numeric value of the input. In contrast, a polynomial time algorithm is polynomial in the size of the input (the number of digits).

Both Knapsack and MCKP have several known approximation algorithms [13], and both have an FPTAS [13]. In addition, both problems have pseudo-polynomial dynamic programming algorithms that find the optimal solution [13]. In particular, there exists a pseudo-polynomial algorithm whose running time is $O(n \cdot c)$, where n is the number of items and c is the knapsack capacity. Such an algorithm is practical only when c is small.

Theorem 1: MaSP can be solved optimally in pseudo-polynomial time by a transformation to MCKP.

Proof: We transform an instance of MaSP to an instance of MCKP such that a feasible solution for MCKP with profit p is also a feasible solution to MaSP with the same profit. Therefore, an optimal solution for MCKP is also an optimal solution for MaSP. The knapsack size is set to $T \cdot C$. For each PHY-profile phy, we define a class of items N_{phy}. We define an item $s_A \in N_{phy}$ for every subset A of PHY-profile phy, where $w(s_A) = H + \sum_{a \in A} w(a)$ and $p(s_A) = \sum_{a \in A} p(a)$. It is easy to see that a feasible solution to MCKP with profit p is also a feasible solution to MaSP with profit p.

At first glance it seems that the time required for this reduction is not polynomial, because the number of subsets is exponential. However, if there are two subsets A and A' such that $w(s_A) = w(s_{A'})$, then only the one with the greatest profit will be selected. There are at most $T \cdot C$ different item sizes, and for each size the item's profit can be found by an $O(n \cdot T \cdot C)$ algorithm for the single knapsack problem. Furthermore, a careful implementation will find the profit for all possible sizes at once, thereby reducing the total running time to $O(T \cdot C \cdot n)$ for each profile. Thus the total running time of MaSP is $O(T^2 \cdot C^2 \cdot n \cdot \text{number-of-PHY-profiles})$. ■

Typical dimensions of an OFDMA frame are $T \approx 20$ and $C \approx 60$. For these values it is reasonable to use the reduction described in Theorem 1. However, for theoretical completeness it is important to note that the same transformation to MCKP can also be performed implicitly, in polynomial time.

Theorem 2: MaSP has an FPTAS; i.e., an approximation scheme whose running time is polynomial in $\frac{1}{\epsilon}$.
The proof is presented in Appendix I.

V. The Micro Scheduling Problem (MiSP)

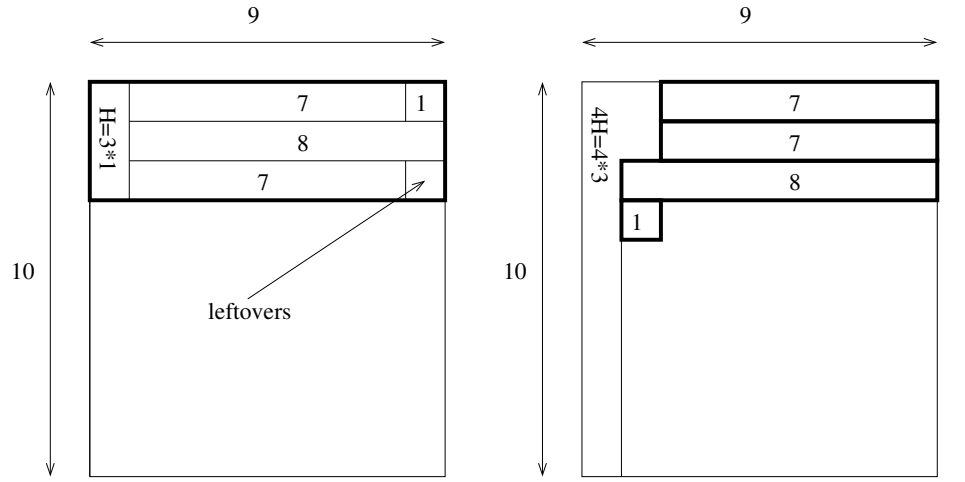
A. Computational analysis of MiSP

After the macro scheduler selects the set of PDUs to be transmitted, the micro scheduler has to build the transmission matrix from rectangles. The scheduler represents each PHY-profile as a sequence of one or more contiguous rectangles, and it faces the following trade-off. It can use many rectangles for each PHY-profile, thereby minimizing the leftovers at the end of each rectangle, it can use the opposite approach of a single rectangle for each PHY-profile, thereby minimizing the header overhead. Figure 4 depicts both approaches for 4 PDUs from the same PHY-profile set, whose sizes are 7, 7, 8, and 1. The first method consumes more space due to the burst overhead $H = 3$, while the second one consumes more space due to leftovers. A good scheduler should find the golden mean between these two opposite approaches.

We now formally define MiSP:

Problem 2 (MiSP):

Instance: An enveloping rectangle $R = [0, W] \times [0, V]$, where W and V are integers, and a set I of n items. Without loss of generality, throughout this section we



(a) one rectangle for each PHY-profile (b) multiple rectangles for each PHY-profile

Fig. 4. The Two Approaches for Micro Scheduling

assume $V \leq W$. For every item $i \in I$, we are given a positive integral size $s(i)$ and a positive profit $p(i)$. The aggregated size of all items does not exceed the size of R, namely, $\sum_{i \in I} s(i) \leq V \cdot W$.

Objective: Find a feasible placement of the items in R with maximum profit. A placement is a pair (I', f), where $I' \subseteq I$ and f is a function from I' to $(R \cap (\mathbb{N} \times \mathbb{N}), \mathbb{N})$. If $f(i) = (p, w)$, then the width of item i is chosen to be w (thus, its height is $\left\lceil \frac{s(i)}{w} \right\rceil$) and its bottom left point in R is located at point p. A feasible placement must fulfill the following requirements: (1) $\forall i \in I'$ the corner points of the rectangle of item i are inside R; (2) $\forall i, j \in I' \wedge \ (i \neq j)$ the rectangles of items i and j do not intersect.

Theorem 3: MiSP is NP-hard. Moreover, unless P=NP, MiSP does not have an FPTAS even in the special case where the profit of a burst is equal to its size, i.e., $\forall i \in I \ s(i) = p(i)$. The proof is presented in Appendix II.

B. A Linear Time Dual Approximation for MiSP

Whereas an approximation algorithm returns a suboptimal solution that meets all the constraints, a dual approximation returns a solution whose profit is at least as high as that of the optimal solution while relaxing some of the constraints. This process is known as resource augmentation. The performance guarantee of a dual approximation is the factor by which the resource is augmented. We use the concept of resource augmentation by relaxing W, the width of the enveloping rectangle R (or V if $V > W$). In order for a dual approximation of MiSP to integrate with MaSP, the matrix size given to MaSP should be smaller than the actual size by some factor.

If the MiSP enveloping rectangle had been continuous, i.e., the slot length had been arbitrarily small, the problem could have been easily solved by setting the width and the length of rectangle i (for burst i) of size $s(i)$ to be V and $\frac{s(i)}{V}$ respectively. Applying this intuition as is to the discrete case would result in a lot of wasted space, since even the smallest item would consume at least V slots. In the following we present an algorithm that captures this rationale yet deals with small profiles without wasting a lot of space. The algorithm is a dual approximation, namely, it can place all items using a space of $(1+f(V))OPT$, where OPT is the space required for optimal placement of all items and $f(V)$ is a $O(V^{-\frac{1}{5}})$. The main idea behind the proposed algorithm is to partition the items into sets in which all items are roughly the same size, and to place each set in separate columns of the enveloping rectangle. This idea is also used in [12] for solving a different variation of the micro scheduling problem[1].

The following is an algorithm for MiSP, with a $(1+f(V))$-approximation guarantee. The value of $f(V)$ is $O(V^{-\frac{1}{5}})$, as discussed later.

Algorithm 1 (An efficient dual approximation for MiSP):

1) Let k be a constant whose value is discussed later.
2) Partition the items into 3 sets P_0, P_1, P_2 in the following way. The set P_0 contains every item whose size is $\leq k$, the set P_1 contains every item whose size is in $(k, k^{\frac{3}{2}}]$, and the set P_2 contains every item whose size is $> k^{\frac{3}{2}}$.
3) For $i = 0, 1$ do
   a) Set $r_i \leftarrow \left\lceil k^{1-2^{-i}} \right\rceil$
   b) Round up the size of each item in P_i to an exact multiple of r_i.
   c) Place P_i in a sequence of r_i consecutive columns, using a next-fit bin-packing algorithm, as presented in [10] (Chapter 2). Each such a sequence is now referred to as a bin.
   Spread out every item in P_i across the entire width of its bin (see Figure 5).
4) Round up all the items in P_2 to an exact multiple of V and place them in some arbitrary order spread out across the entire height of the enveloping rectangle.

Theorem 4: Algorithm 1 is a $(1 + f(V))$-approximation. More specifically, for $k = V^{\frac{4}{5}}$ it has a $(1 + O(V^{-\frac{1}{5}}))$-approximation performance guarantee.

Proof: We need to show that the items can be fitted with an overhead of only a small fraction of the space they occupy. In the worst case, the total size of the items is tight, namely, $V \cdot W = \sum_{i \in I} s(i)$. Compared to the impractical continuous

[1]Our work, which was originally reported in [4], was conducted independently of [12].

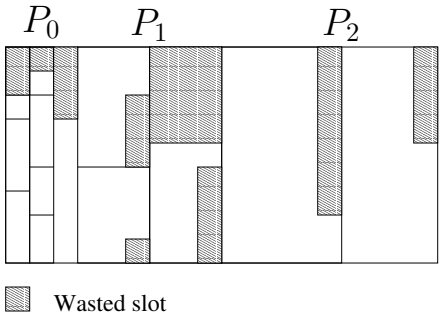


Fig. 5. An example of Algorithm 1, with $k = 9$

algorithm which has no "wasted space", the wasted space of Algorithm 1 stems from steps 3(b), 3(c), and 4. We now analyze the contribution of each of these components to the total wasted space.

In step 3(b), space is wasted because the size of the item is rounded up to an exact multiple of $r_i = \left\lceil k^{1-2^{-i}} \right\rceil$. Thus, each item consumes at most $r_i - 1$ more slots than its size. For P_0, there is no wasted space because $r_i - 1 = 1 - 1 = 0$. For P_1 the size of the smallest item is at least k and it is rounded to a multiple of $\left\lceil \sqrt{k} \right\rceil$. Therefore, the total fraction of wasted space is $\left(\left\lceil \sqrt{k} \right\rceil - 1 \right) / k < \sqrt{k}/k = 1/\sqrt{k}$.

In step 3(c) space is wasted because not all the bins are completely filled. Note that the height of each item in P_i is at most k, because the maximum size of any item in P_i is $k^{2-2^{-i}}$, and each item is spread out across $r_i = \left\lceil k^{1-2^{-i}} \right\rceil$ consecutive columns. Therefore, all the bins except at most one are at least $(V - k)$-full. Since the size of each bin is V, the fraction of wasted space is $\frac{k}{V}$. For the last bin the wasted space is at most 100%, namely, $r_i \cdot V$. The total wasted space due to the last bin of step 3(c), for all iterations, is therefore $V \cdot (r_0 + r_1) = V \cdot (1 + \left\lceil \sqrt{k} \right\rceil)$. The wasted space is bounded by $\frac{V(1+\lceil \sqrt{k} \rceil)}{V \cdot W} = \frac{1+\lceil \sqrt{k} \rceil}{W}$ of the total space $V \cdot W$. Since $V \leq W$, the total wasted space is bounded by $\frac{1+\lceil \sqrt{k} \rceil}{V}$.

In step 4, space is wasted because the size of every item is rounded up to an exact multiple of V. The minimum size of any item in P_2 is at least $k^{3/2}$, and each item wastes at most V slots. Therefore, the fraction of wasted space is $\frac{V}{k^{3/2}}$.

To summarize, the total approximation guarantee $A(V, k)$ is bounded by:

$$A(V,k) \leq k/V + \left(1 + \left\lceil \sqrt{k} \right\rceil \right) / V + 1/\sqrt{k} + V/k^{\frac{3}{2}}.$$

This function can be minimized for $k = V^{\frac{4}{5}}$, in which case we get $A(V, V^{\frac{4}{5}}) = O(V^{-\frac{1}{5}})$. ■

Theorem 5: The running time of Algorithm 1 is linear in the number of items.

Proof: Steps 2 and 4 can be performed in a constant time per item. The linear running time of step 3(c) follows from [10] (Chapter 2). ■

The next-fit bin-packing algorithm was good enough to achieve the approximation ratio. However, in practice, algorithms that outperform next-fit can improve the performance. One of these algorithms is the best-fit decreasing algorithm [10] (Chapter 2), which runs in $O(n \log n)$.

In Appendix III we present a dual PTAS for MiSP that uses Algorithm 1 as a subroutine. Using this algorithm, one can also derive a constant approximation (not just a dual approximation) for MiSP by extracting a feasible solution from a super optimal dual solution. Using partial enumeration and the dual MiSP, these techniques can also deliver a PTAS for MiSP.

C. Heuristics for MiSP

Having presented an algorithm with worst-case performance guarantee, we now present several heuristics for MiSP. While these algorithms might not perform very well in the worst case scenario, their performance is very good in practice, as shown in Section VIII.

The first algorithm is called Increasing Size. The rationale behind this algorithm is to schedule items of roughly the same size next to each other. A variation of this idea was used in Algorithm 1. However, while in Algorithm 1 the boundary between two sets is fixed, the Increasing Size algorithm tries to set it dynamically.

Algorithm 2 (An increasing size algorithm for MiSP):

1) Maintain the PHY-profiles as a list L sorted by their size in increasing order.
2) While there are PHY-profiles left in L do
   a) For every k and t, check the relative space wasted when scheduling the first t PHY-profiles that fit on bins of k consecutive rows. This is the classic bin-packing problem of the first t PHY-profiles where each bin contains k consecutive rows and each item is spread out across the entire height of the bin. The space used, S, is $k \cdot T$ times the number of bins used, so the relative space wasted is S over the sum of sizes of the first t PHY-profiles.
   b) Select the PHY-profiles that produce the minimum relative wasted space.
   c) Remove the selected items from L, i.e., the first t items for the choice of k and t that minimized the space wasted.

Claim 1: The maximum value of k in Algorithm 2 is bounded by $2C$.

Proof: From Algorithm 1 and Theorem 4 we know that there exists a schedule such that the number of rows is at most $C \cdot (1+T^{-\frac{1}{5}})$. We expect Algorithm 2 to outperform Algorithm 1, but one can see that every choice of $k > C \cdot (1+T^{-\frac{1}{5}})$ clearly performs worse. Therefore, the maximum value of k in which this heuristic outperforms Algorithm 1 is bounded by $2C$. ■

Let n be the number of PHY-profiles. It is easy to see that $t \leq n$, and that the total number of choices of k and t for a single iteration is bounded by $2C \cdot n$. The running time for applying the best-fit decreasing algorithm for the bin-packing problem is $O(n \log n)$. Therefore, the total running time for a single iteration of the algorithm is $2C \cdot n \cdot n \log n$, and the total running time $O(2C \cdot n^3 \log n)$. Since a typical value of n is smaller than 10, and a typical value of C is smaller than 100, this running time can be considered reasonable.

The next algorithm is called Decreasing Size. The rationale behind this algorithm is the same as for Increasing Size. However, PHY-profiles are considered in reverse order, so that the asymptotic running time will decrease when the best-fit decreasing bin-packing algorithm is used.

Algorithm 3 (A decreasing size algorithm for MiSP):

Same as Algorithm 2, except that in step 1 the items are sorted in decreasing order.

The running time of Algorithm 3 is similar to that of Algorithm 2. However, as t grows, we can use the result of the previous iteration because the new PHY-profile has a smaller size. This reduces the total running time to $O(2C \cdot n^2 \log n)$.

The last algorithm we present is called Best Fit. The intuition behind this algorithm is to dynamically choose items that locally minimize the wasted space. In the previous algorithms this was achieved by selecting items that are roughly the same size. Here we try to achieve this goal by optimizing the schedule on k consecutive rows, as in Algorithm 1. But unlike Algorithm 1, this algorithm tries all possible combinations of PHY-profiles, and picks the best one.

Algorithm 4 (A best fit algorithm for MiSP):

1) Add all PHY-profiles to a list L.
2) While there are PHY-profiles left in L do
   a) For every k, check the relative space wasted when scheduling any PHY-profiles with a fixed height of k. This is a knapsack problem, where items are spread out across the entire height of the knapsack. The relative space wasted is $k \cdot T$ over the sum of sizes of the scheduled items.
   b) Select the PHY-profiles that produce the minimum relative wasted space.
   c) Remove the selected items from L.

The total running time of Algorithm 4 is $O(2C \cdot T \cdot n^2)$, where n is the number of PHY-profiles, because Claim 1 applies here as well. The running time of single iteration is, therefore, $2C \cdot T \cdot n$, and the total running time is $O(2C \cdot T \cdot n^2)$. Since a typical value of T is ≤ 40, this running time can be considered reasonable.

VI. The Extended Macro Scheduling Problem (E-MaSP)

As discussed in Section III, another approach for MaSP is not to decide in advance (off-line) what the PHY-profile of every PDU should be, but rather to make this decision on-line, as part of MaSP. As far as we know, this idea has never been considered before. The resulting optimization problem, referred to as E-MaSP (Extended MaSP), is defined as follows:

E-MaSP: An Extended Macro Scheduling Decision: Deciding which PHY-profiles will be used for every PDU, which PHY-profiles will be accommodated in the next frame, and which PDUs will be transmitted for every selected PHY-profile.

An E-MaSP-based scheduler has advantages over a MaSP-based scheduler especially when the channel is not heavily loaded. As an example, consider a single PDU waiting for transmission. The MaSP-based scheduler will choose to schedule this PDU using a pre-determined PHY-profile. In contrast, due to the availability of bandwidth, the E-MaSP-based scheduler will transmit this PDU using the most robust PHY-profile. Hence, the profit gained by the E-MaSP scheduler in this case is likely to be higher than the profit gained by the MaSP scheduler, and this extra profit is gained at no additional cost.

We now formally define E-MaSP:

Problem 3 (E-MaSP):

Instance: The frame's size $L = T \cdot C$, the overhead H for using a PHY-profile, a set $P = \{p_1, p_2, \ldots, p_n\}$ of PDUs awaiting transmission, and a set $PHY = \{PHY_1, PHY_2, \ldots, PHY_m\}$ of PHY-profiles. Each pair of PDU p_i and PHY-profile PHY_j is associated with a non-negative profit $p(PHY_j(p_i))$ and non-negative size $s(PHY_j(p_i))$. These profit and size are relevant when the considered PDU is transmitted using this specific PHY-profile. In addition, each PHY-profile has a fixed overhead H.

Objective: Find a feasible schedule S with maximum profit. A schedule is a subset P' of PDUs and an assignment function f from P' to PHY. The size of a schedule S is equal to H times the number of PHY-profiles used by S, plus the cumulative size of the accommodated PDUs (recall that the size for each PDU depend upon the PHY-profile selected for it), namely, $H \cdot |\cup_{p_i \in P'} f(p_i)| + \sum_{p_i \in P'} s(f(p_i))$. A feasible schedule is a schedule whose size is $\leq L$. The profit of a subset S is the cumulative profit gained by the scheduled PDUs with respect to the PHY-profile selected for each of them, namely, $\sum_{p_i \in P'} p(f(p_i))$.

Note that in some cases a PDU is destined to a user that does not support certain PHY-profiles. This case can be captured by E-MaSP by assigning a 0 profit to the combination of this PDU and each such PHY-profile.

In Sections III and IV we showed that both MaSP and MiSP are NP-hard but can be approximated with arbitrarily close precision in polynomial time (see Figure II). In the following we show that E-MaSP is not only NP-hard, but also cannot be approximated with a factor smaller than $\frac{e}{e-1}$. This implies that E-MaSP is computationally harder than MaSP and MiSP.

Theorem 6: E-MaSP is NP-hard. Moreover, it cannot be approximated within a factor small than $\frac{e}{e-1}$.
The proof is presented in Appendix IV.

An observation from the proof of Theorem 6 is that E-MaSP is a generalization of the Budgeted Maximum Coverage Problem [14]. The latter is a generalization of the Maximum Coverage Problem, in which the S_i subsets have arbitrary costs.

A key difference between Budgeted Maximum Coverage and E-MaSP is that even if we know which PHY-profiles should be used in the optimal schedule, we still have to find a valid assignment of PDUs to these PHY-profiles, since the PDUs have a non-negative size and non-negative profit for each PHY-profile. Therefore, an algorithm must also find an assignment of PDUs to the selected PHY-profiles. In addition, it must find a way to treat PDUs that were selected to PHY-profiles with low profit and low size. Such PDUs should be reconsidered for an "upgrade" to a PHY-profile with a higher profit and size.

In the following we present an approximation algorithm for E-MaSP, which is a generalization of the greedy algorithm proposed in [14] for Budgeted Maximum Coverage. The main idea behind the new algorithm is to define the residual profit of PDUs that have already been selected by the scheduler as the original profit minus the profit gained when the PDU was previously selected. The rationale behind this is that by selecting a PDU for one PHY-profile instead of the other we gain only the profit difference and pay only for the size difference. The algorithm iteratively picks up the best combination of a PHY-profile and a set of PDUs assigned to it, until the frame's capacity $L = T \cdot C$ is reached. When this greedy selection ends, its output is compared to the most profitable schedule of a single PHY-profile. From these two schedules, the one with the highest profit is chosen.

Algorithm 5 (A greedy algorithm for E-MaSP):

1) Let S be an empty schedule.
2) While changes are made to S do:
   a) If there is a PHY-profile ϕ that has already been selected to S and a PDU m such that (a) the residual size of assigning m to ϕ is not positive, and (b) the residual profit of assigning m to ϕ is positive, then add m to S and assign it to ϕ.
   b) Otherwise, if there is a PHY-profile ϕ and a set of PDUs M such that (1) the residual size of assigning any PDU from M to ϕ is non-positive, (2) the residual profit of assigning any PDU from M to ϕ is positive, and (3) the total size of adding ϕ and M to S while assigning all the PDUs from M to ϕ is non-positive, then add ϕ and M to S and assign all the PDUs in M to S.
   c) Otherwise, select a PHY-profit ϕ and a set of PDUs M, such that by adding ϕ to S and assigning all the PDUs in M to ϕ we achieve the highest residual density (the residual density is the residual profit divided by the residual size). Add ϕ to S and assign all the PDUs from M to S.
3) Find a maximum profit schedule with exactly one PHY-profile. If the profit of this schedule is greater than the profit of S, as found in the previous steps, return this schedule. Else, return S.

During each iteration of the algorithm at least one PDU is chosen. Therefore, the combination of this PDU and the PHY-profile to which it is chosen will not be considered again. Thus, Algorithm 5 has at most $n \cdot m$ iterations, where n is the number of PDUs and m is the number of PHY-profiles. Although finding a set with maximum density, as required in step 2(c), is NP-hard, such a set can be found by a pseudo polynomial algorithm. Since L is in the order of 1200, using a pseudo polynomial algorithm is reasonable. If L is too large, an $(1+\epsilon)$-approximation can be applied and it will only affect the approximation analysis by $(1+\epsilon)$.

Theorem 7: Algorithm 5 is a $(\frac{2e-1}{e-1})$-approximation[2] for E-MaSP. Moreover, this algorithm can be implemented in $O(m^2 \cdot n^2 \cdot L)$ where n is the number of PDUs, m is the number of PHY-profiles, and L is the size of the frame.

Proof: Algorithm 5 is a special case of the algorithm we presented in [6], for which this performance guarantee is proven. ∎

Algorithm 5 can be extended using partial enumeration to get an $e/(e-1)+\epsilon$ performance guarantee, which is the best possible up to an ϵ.

VII. Multi frame optimization

The discussion so far addresses the optimization of a single frame. However, in practice one would be more interested in optimizing the performance of the whole system. In particular, maximizing the profit from a sequence of consecutive frames ("global optimization") is more important than maximizing the profit of a single frame ("local optimization"). An optimal local solution does not necessarily yield an optimal global solution. For example, consider the case where a PDU has a small profit in the current frame and a higher profit in a future frame. This happens if the channel conditions for the destination host are momentarily bad. Assuming there are no other pending PDUs, the local optimizer would schedule this PDU in the current frame, whereas a global optimizer would leave this PDU for a future frame.

To simplify discussion, consider the most simple model for a local optimizer, where the OFDMA matrix has only one logical channel and only one PHY-profile. In this case, the local optimization can be viewed as a plain Knapsack problem. The corresponding global optimization problem can be viewed as the Generalized Assignment Problem (GAP), defined as follows [3]:

Instance: A pair (B, S) where B is a set of m bins (knapsacks) and S is a set of n items. Each bin $b \in B$ has capacity $c(b)$. For each item i and bin b, $s(b,i)$ and $p(b,i)$ are the size of item i in bin b and the profit of item i in bin b respectively.

Objective: Find a feasible assignment with maximum profit. An assignment is a function f from S to $B \cup \{NIL\}$. A feasible assignment is an assignment such that $\forall_{b \in B} \sum_{i|f(i)=b} s(b,i) \leq c(b)$. The profit of an assignment is $\sum_i p(f(i), i)$, where $\forall_i p(nil, i) = 0$.

The sequence of OFDMA frames are represented by the bins, the PDUs are represented by the items, and a PDU has a different profit $p(b,i)$ and size $s(b,i)$ for each frame. A different size is possible because different modulations might be considered for the same PDU in different frames.

Even in this simple case, the global optimization problem is computationally much harder than the local optimization problem. This is because, unlike Knapsack, GAP does not have an FPTAS or even a PTAS [3]. Therefore, a constant approximation algorithm is theoretically the best possible.

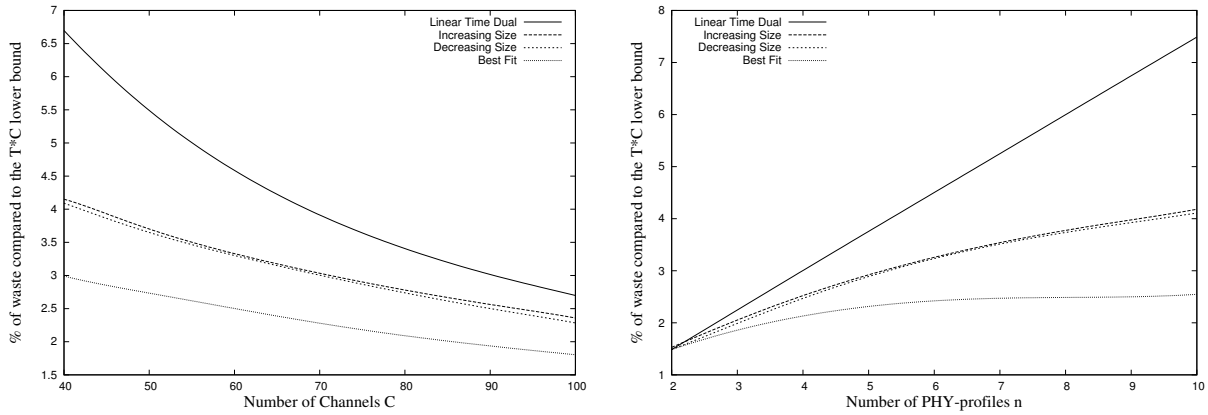
In [7], we present a generic approach to transform a local optimization algorithm A into a global optimization algorithm B under the following conditions: (1) if S is a feasible solution then any subset of S is also a feasible solution; and (2) the objective is to maximize the total aggregated profit. If A is an α-approximation for the local optimization problem, then the approach from [7] guarantees that B is a $(1+\alpha)$-approximation for the global problem. The running time of B for n frames is n times the running time of A. Since both of these conditions hold in our OFDMA model, this approach can be used to provide a global optimization for multiple frames.

VIII. Simulation Study

The purpose of this section is two-fold. First, to compare the performance of the MiSP algorithms presented in Section V. Second, to compare between the performance of the MaSP and the E-MaSP algorithms. In the following graphs, the performance is measured against the $T \cdot C$ lower bound, namely, the actual size of the PHY-profiles. We measure the wasted space used by every algorithm: we view the extra space as the difference between the actual size of PHY-profiles and $T \cdot C$ and the ratio between the extra space and $T \cdot C$ as the overhead (space wasted). During the simulation study, we consider a Gaussian distribution for the PHY-profile sizes, with a mean of $\frac{T \cdot C}{n}$. With this distribution, all PHY-profiles are roughly the same size. Simulations under different distributions, such as uniform, or under different means, show similar results. Note that typical values of C, T, and n are $35 \leq C \leq 70$, $13 \leq T \leq 26$, and $3 \leq n \leq 8$.

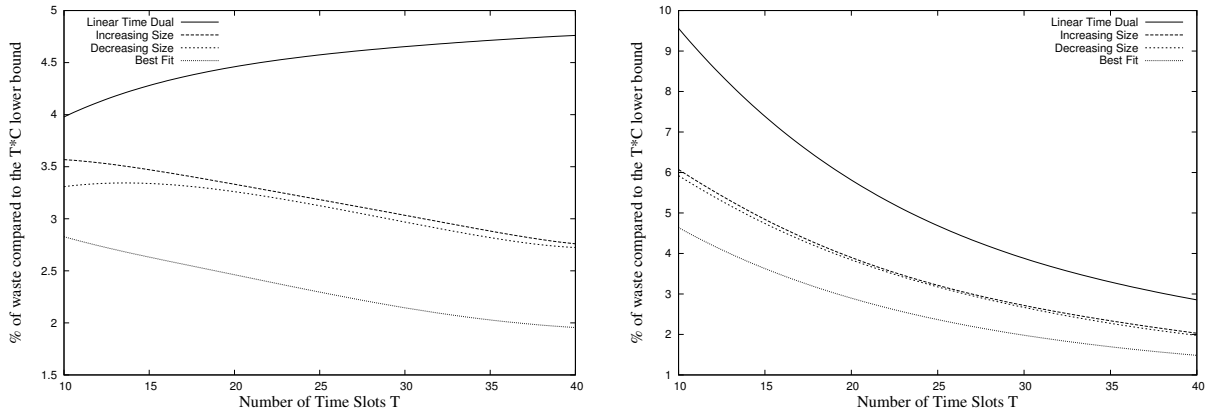
Figure 6(a) shows the effect of C on the performance, assuming $T = 20$ and $n = 6$. Figure 6(b) shows the effect

[2] $(2e-1)/(e-1) \approx 2.58$



(a) The effect of C ($T = 20$ and $n = 6$)

(b) The effect of n ($T = 20$ and $C = 60$)



(c) The effect of T ($C = 60$ and $n = 6$)

(d) The effect of $\frac{C}{T}$ ($C = 2.5T$ and $n = 6$)

Fig. 6. The Performance of the Algorithms as a Function of Various Parameters.

of n on the performance, assuming $T = 20$ and $C = 60$. Figure 6(c) shows the effect of T on the performance, assuming $C = 60$ and $n = 6$, and Figure 6(d) shows the effect of $\frac{C}{T}$ on the performance, assuming $C = 2.5T$ and $n = 6$.

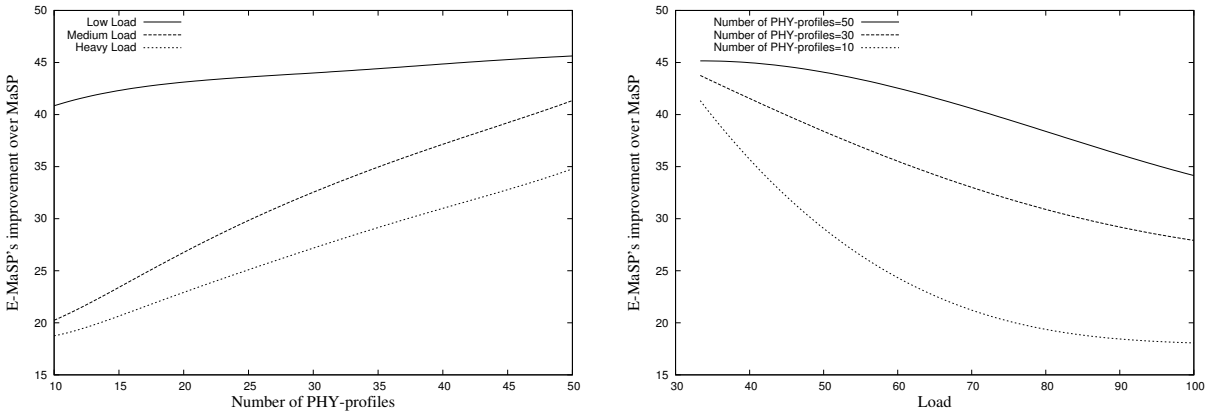
From Figure 6(a) we can see that when C increases, the performance of all the algorithms improve. This is because when C and the PHY-profiles are bigger, the matrix can be viewed as continuous, in which case the produced schedules are closer to optimal.

In Figure 6(b), all the algorithms have similar performance for $n = 2$. This is because the ways in which such a small number of profiles can be scheduled are severely limited. As n grows, the Best Fit algorithm has more combinations of PHY-profiles to choose from (e.g. $2^n - 1$ in the first iteration), while Increasing Size and Decreasing Size have only n in their first iteration, and Dual has only 1. The difference in the number of combinations between Best Fit and the rest of the algorithms stems from the loose structure of the scheduled items, i.e., the largest and smallest items can be scheduled next to each other. Increasing Size and Decreasing Size perform roughly the same, since they have the same number of combinations to choose from. Dual has only 1, and its performance decreases linearly with n. More specifically, when T and $\frac{TC}{n}$ are relatively large, the expected wasted space per item is $\frac{T}{2}$, and the total expected wasted space is $\frac{nT}{2TC} = \frac{n}{2C}$. In Figure 6(b), where $T = 20$ and $C = 60$, we get $\frac{n}{2 \cdot 60} = \frac{n}{120}$.

From Figure 6(c) we can see that when T increases, the performance of all the algorithms except Dual improves. The Dual algorithm's performance declines when T grows because, for a high value of T, all items are relatively large, and are rounded to an exact multiple of T. When T is small, there is a greater chance for a PHY-profile size to be an exact multiple of T. When T increases, the expected space wasted due to this rounding is $\frac{T}{2}$ per item. Therefore, the total wasted space when T is relatively large converges to $\frac{nT}{2CT} = \frac{n}{2C}$. For the parameters in Figure 6(c), i.e., $C = 60$ and $n = 6$, we get $\frac{6}{2 \cdot 60} = 5\%$. For all the other algorithms, the improvement is due to the greater chance that two or more items can be fit into a single bin. Again, the number of combinations of PHY-profiles to choose from plays a critical role.

Finally, Figure 6(d) shows that the algorithms are highly scalable, because their performance improves when T and C increase, which indicates that MiSP becomes easier to solve. For the Increasing Size, Decreasing Size, and Best Fit algorithms, this improvement is attributed to both the increase in T (see Figure 6(c)) and the increase in C (see Figure 6(a)). For the Dual algorithm, the improvement stems from our earlier observation on Figure 6(c), where the performance converges to $\frac{n}{2C}$ when T increases.

We conclude this section by showing the advantage of E-MaSP over MaSP. Although the proposed MaSP algorithm was shown to be optimal, it can only assign each PDU to a single, pre-determined, PHY-profile. In contrast, E-MaSP can assign



(a) The effect of the number of PHY-profiles

(b) The effect of the load

Fig. 7. The Relative Performance Improvement of E-MaSP vs MaSP.

a PDU to every profile, but each assignment is associated with a different profit and cost. Therefore, the performance of E-MaSP is expected to be better than the performance of MaSP. For the following simulation study, we consider a Gaussian distribution of PDU sizes. In addition, we set $C = 60$, $T = 20$, and a burst overhead of 5%. All physical channel aspects are translated into an SNR, which, along with PHY-profile, is translated into the packet loss probability. We assume that the MaSP scheduler chooses the most cost-effective PHY-profile for each PDU. That is, the PHY-profile for which the probability for successful transmission divided by the bandwidth cost is maximum.

Figure 7(a) shows the throughput improvement of E-MaSP over MaSP as a function of the number of PHY-profiles, for low (33%), medium (66%), and heavy (100%) loads. We can see that the improvement of E-MaSP increases when the number of PHY-profiles increases. The reason for this is that the MaSP algorithm might select two high profit PDUs for different PHY-profiles with close parameters. This increases the overhead penalty without increasing the likelihood for successful transmission. In contrast, E-MaSP tends to minimize the number of PHY-profiles when the effect on the loss probability is negligible. Therefore, MaSP is likely to accommodate more PHY-profiles in each frame, and, therefore, to increase the bandwidth overhead and the bandwidth loss due to fragmentation.

Figure 7(b) shows explicitly the correlation between the load and the E-MaSP's improvement. We can see that the improvement is much higher for low loads. This is because MaSP chooses the PHY-profiles without taking the load into account. In contrast, E-MaSP upgrades PDUs to more robust modulation schemes when there is enough available space.

IX. Conclusions

In this paper we defined and studied the intricate problem of downlink scheduling on an OFDMA channel. The main contributions of this paper are breaking the OFDMA scheduling problem into two more tractable problems, referred to as macro scheduling (MaSP) and micro scheduling (MiSP), analyzing the computational complexity of these two problems, and developing efficient algorithms for solving them.

We showed that MaSP can be solved optimally in pseudo-polynomial time by transforming it into the Multiple Choice Knapsack Problem, and that it can be approximated with an arbitrarily good precision in polynomial time. Then, we showed that MiSP is a more difficult problem than MaSP, in the sense that a $(1+\epsilon)$-approximation scheme for this problem cannot run in in polynomial time in $\frac{1}{\epsilon}$. Nevertheless, we presented several efficient algorithms for MiSP.

We also presented an extended version of the macro scheduling problem, called E-MaSP. In this version, the association between a PDU and its PHY-profile is determined on-line by the scheduler, as part of the macro scheduling. We showed that E-MaSP has an advantage over MaSP especially when the channel is underloaded, because in such a case the probability for a successful transmission of a PDU using the PHY-profile determined by E-MaSP is greater than when the PHY-profile is determined in advance regardless of the load in the channel. However, the performance gains achieved by E-MaSP comes at the expense of higher running time complexity.

One of our most important results was that the Best Fit algorithm performs better than the other MiSP algorithms. In particular, with typical parameters, this algorithm wastes only 2.5% of the total scheduling space.

Appendix I
Proof of Theorem 2

To prove this theorem, we first describe a simple FPTAS for MCKP. Although this is not the algorithm with the best running time, it is probably the simplest one.

Let $M[i,j]$ be a table of size $m \times P$, where m is the number of classes of items, and P is the sum of the profits of all items. Let $M[i,j]$ be the minimum size knapsack with profit gain j from the $N_1, N_2, \ldots, N_i$ classes, and let n be the number of items. Using dynamic programming, M can be filled in $O(n \cdot P)$ time. In order to solve MCKP in polynomial time, we scale all profits by a factor of $\frac{n}{\epsilon \cdot P}$, and then calculate M using the new scaled profits. The returned solution must be within a $(1+\epsilon)$ factor from the optimum [13]. Note that if there are two items with the same profit, the above algorithm selects the one with the lowest size.

The algorithm uses profit scaling and therefore is concerned only with a polynomial number n/ϵ of different profits. Hence, the reduction can be performed implicitly, in the following way. Whenever the above algorithm seeks an item with a specific scaled profit, it chooses the minimum weight item with this profit gain. Therefore, an approximation that uses profit scaling can be implemented in polynomial time. Specifically, it

takes $O\left(n^3/\epsilon\right)$ to compute the profits and weights for a single PHY-profile and $O\left(n^3/\epsilon^2 \cdot \text{number-of-PHY-profiles}\right)$ in total. Since this is basically the same reduction, a feasible solution for MCKP with profit p is also a feasible solution to MaSP with the same profit. Thus, an α-approximation for MCKP is also an α-approximation for MaSP.

APPENDIX II
PROOF OF THEOREM 3

We first present a known NP-complete problem, called Equipartition [8]:

Instance: A set I of $2m$ items. Each item i has a positive integral size $s(i)$, and $\sum_{i\in I} s(i) = 2S$.

Objective: Find a subset $I' \subseteq I$ of items such that $\sum_{i\in I'} s(i) = \sum_{i\in I\setminus I'} s(i) = S$, and $|I'| = m$.

We reduce Equipartition to MiSP. Each Equipartition item i is transformed into an MiSP item i^*. Both weight and profit of item i^* are set to $2(s(i)+S)+1$. This guarantees that all sizes are odd. We also set $V = 2$ and $W = 2S + 2mS + m$. We call each (i,j)-entry in the enveloping rectangle a slot.

If there is a valid equipartition of I, then all items can be placed in the two different rows, such that each row contains exactly m items. Since all sizes are odd, no item can be spread over two rows without wasting a slot. Moreover, no row may contain more than m items, since the sum of weights of each set of $m+1$ or more items is at least $(m+1)(2S+1) = 2Sm + 2S + m + 1 > 2S + 2Sm + m$. This implies that if no valid equipartition exists, the optimal MiSP solution cannot contain all items and its maximum profit is at most $2S+4Sm+2m > 4S + 4Sm + 2m - 2S - 1$.

We now show how to use an FPTAS for MiSP to solve Equipartition. Suppose there is an FPTAS for MiSP, namely, a $(1+\epsilon)$-approximation scheme whose running time is polynomial both in n and $1/\epsilon$. When an equipartition exists, the optimal profit of MiSP is $4S + 4Sm + 2m$. When an equipartition does not exist, the profit of MiSP is at most $2S + 4Sm + 2m$. By setting

$$\epsilon = 1/\left(1+3m\right) = 2S/\left(2S+4Sm+2Sm\right) < 2S/\left(2S+4Sm+2m\right) = \left(4S+4Sm+2m\right)/\left(2S+4Sm+2m\right) - 1,$$

we can distinguish between these two cases. If the running time is polynomial in $\frac{1}{\epsilon} = 3m+1$, we can solve Equipartition in polynomial time, which implies that P=NP.

APPENDIX III
A DUAL PTAS FOR MISP

Algorithm 1 has a fixed performance guarantee that decreases as V grows, and a linear running time. We now present Algorithm 6, a dual PTAS for the MiSP. This algorithm is impractical due to its very high running time. However, it is presented for the sake of theoretical completeness. The algorithm produces a dual $(1+\epsilon)$-approximation for every $\epsilon > 0$. Here we distinguish between two values of ϵ. We say that ϵ is big if $\epsilon > f(V)$, where $f(V)$ is the performance guarantee of Algorithm 1, and it is small otherwise. When ϵ is big, we can use Algorithm 1 to provide an approximate solution. If ϵ is small, we again divide the solution into two cases according on the size of W. We say that W is small if $W < \frac{4V}{\epsilon^2}$ and it is big otherwise. If W is small, we use an optimal dynamic programming subroutine. If W is big, we treat small items as jobs to be scheduled on parallel machines (rows) to minimize the makespan, and we treat big items as we did in Algorithm 1.

Algorithm 6 (A dual PTAS for MiSP):

1) If $f(V) < \epsilon$, run Algorithm 1 and return its solution.
2) Else,
   a) If $W \leq \frac{4V}{\epsilon^2}$, solve the problem optimally using dynamic programming.
   b) Else, partition the items into two sets of big and small items. The set of big items, P_b, holds every item whose size is $> \frac{2V}{\epsilon}$. The set of small items, P_s, holds the rest of the items.
   c) Treat each item in P_b in the same way items in P_m are treated in step 4 of Algorithm 1. Namely, round it to an exact multiple of V, and place it in some arbitrary order spread out across the entire height dimension.
   d) Each item in P_s is placed in a single row. Each of the V rows is considered as a machine and each item as a job with the same size. Then, the greedy algorithm from [10] (Chapter 9) is invoked to schedule the jobs on the machines.

Theorem 8: The running time of Algorithm 6 is polynomial in the number of items.

Proof: By Theorem 5, step 1 can be implemented in linear time. It is easy to verify that steps 2(b)–2(d) can be implemented in polynomial time. To implement step 2(a) we use a dynamic programming procedure. For specific values A and B, this procedure decides if all the items fit in $R' = [0,A]\times[0,B]$. In iteration i we examine all possible placements of item i. At the end of the n'th iteration, we can tell if all items have been placed. The number of possible configurations for each item is at most $A^2 \cdot B$, since there are at most $A \cdot B$ possible placements of the upper left corner and at most A choices for the height. Since there are $2^{A\cdot B}$ configurations of R', the total complexity is $O(n \cdot A^2 \cdot B \cdot 2^{A\cdot B})$. When the dynamic programming procedure is used, W and V are small, namely, $\epsilon \leq f(V)$ and $W < \frac{4V}{\epsilon^2}$. Thus, this procedure runs in polynomial time in n, but is exponential in $\frac{1}{\epsilon}$. By Algorithm 1, all items fit in $R' = [0, W + Wf(V)] \times [0, V]$. Therefore, the maximum value of A and B are polynomial in V and W. ∎

Theorem 9: Algorithm 6 is a dual $(1+\epsilon)$-approximation.

Proof: In the worst case, the total size of all the items is tight, namely, $V \cdot W = \sum_{i\in I} s(i)$. The performance guarantee of step 1 stems from Theorem 4, and that of step 2(a) from the optimality of the dynamic programming procedure. In step 2(c) the big items are rounded up by at most V. Therefore, the total space wasted for big items is $< \frac{V}{2V/\epsilon} = \frac{\epsilon}{2}$. In step 2(d), at most V rows waste space of $2V/\epsilon$, and the relative wasted space is $< \frac{2V^2/\epsilon}{WV} = \frac{2V}{W\epsilon}$. Because $W > \frac{4V}{\epsilon^2}$, the relative wasted space is $< \frac{2V\epsilon^2}{4V\epsilon} = \frac{\epsilon}{2}$. Therefore, the total relative wasted space in step 2(c) and 2(d) is $< \epsilon$. ∎

APPENDIX IV
PROOF OF THEOREM 6

We first present the well-known NP-hard Maximum Coverage problem [6]:

Instance: A number k and a collection of sets $S = S_1, S_2, \ldots, S_m$, where $S_i \subseteq \{x_1, x_2, \ldots, x_n\}$.

Objective: Find a subset $S' \subseteq S$ of sets, such that $\left|S'\right| \leq k$ and the number of covered elements $\left|\bigcup_{S_i \in S'} S_i\right|$ is maximized.

We now show a reduction from Maximum Coverage to E-MaSP. Each item x_i in the Maximum Coverage instance is transformed into a PDU. Each set S_j is transformed into a PHY-profile. For an item x_i and a set S_j, if $x_i \in S_j$ then the profit and size of the assignment of the corresponding PDU to the corresponding PHY-profile are 1 and 0 respectively. If $x_i \notin S_j$, the profit and size are 0 and ∞ respectively. In addition, the overhead H for every PHY-profile is set to 1, and the frame size is set to k.

It is clear that at most k PHY-profiles can be scheduled in the next frame. For the scheduled PHY-profiles, it is clear that the total profit is not higher than the number of PDUs that can be assigned to the selected PHY-profiles. Therefore, a schedule that gains profit of p is translated into a maximum coverage solution whose size is $\leq k$ that covers at least p items. This proves that an optimal schedule gains no more than optimal maximum coverage.

On the other hand, a maximum coverage solution of size k that covers m elements can be transformed into a schedule whose size is k and that schedules m PDUs. This is done by choosing the corresponding PHY-profiles and assigning a (covered) PDU to one of them. Therefore, an optimal schedule gains as much as the optimal maximum coverage.

Since an optimal solution for maximum coverage can be translated (in polynomial time) into an optimal solution for the specific E-MaSP instance and vice versa then: (a) the fact that Maximum Coverage is NP-hard implies that E-MaSP is also NP-hard; (b) the fact that Maximum Coverage cannot be approximated within a factor smaller than $\frac{e}{e-1}$, under standard assumptions[3] [14], implies that E-MaSP also cannot be approximated within the same factor.

REFERENCES

[1] A. Azgin and M. Krunz. Scheduling in wireless cellular networks under probabilistic channel information. In *ICCCN*, pages 89–94, 2003.

[2] Y. Ben-Shimol, I. Kitroser, and Y. Dinitz. Two-dimensional mapping for wireless ofdma systems. *IEEE Trans. Broadcasting*, 52(3):388–396, Sept 2006.

[3] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

[4] R. Cohen and L. Katzir. Computational analysis and efficient algorithms for micro and macro ofdma scheduling. Technical Report CS-2007-02, Technion-Israel Institute of Technology, April 2007.

[5] R. Cohen and L. Katzir. A generic quantitative approach to the scheduling of synchronous packets in a shared uplink wireless channel. *IEEE/ACM Trans. Netw.*, 15(4):932–943, 2007.

[6] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Inf. Process. Lett.*, 108(1):15–22, 2008.

[7] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Inf. Process. Lett.*, 100(4):162–166, 2006.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[9] A. J. Goldsmith and S. G. Chua. Variable-rate variable-power MQAM for fading channel. *IEEE Transactions on Communications*, 45(10), Oct. 1977.

[10] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[3] $NP \nsubseteq DTIME(n^{O(\log\log n)})$, namely, that not all problems in NP can be solved in $O(n^{c\log\log n})$ for every constant c.

[11] Institute of Electrical and Electronics Engineers Inc. IEEE standard for local and metropolitan area networks – part 16: Air interface for fixed broadband wireless access systems, 2004.

[12] A. Israeli, D. Rawitz, and O. Sharon. On the complexity of sequential rectangle placement in ieee 802.16/wimax systems. In *ESA*, pages 570–581, 2007.

[13] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[14] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[15] D. Kivanc, G. Li, and H. Liu. Computationally efficient bandwidth allocation and power control for OFDMA. *IEEE Transactions on Wireless Communications*, 2(6), Nov. 2003.

[16] S. Lu, V. Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, 7(4):473–489, 1999.

[17] S. Nanda, K. Balachandran, and S. Kumar. Adaptation techniques in wireless packet data services. *IEEE Communications Magazine*, 38(1), Jan. 2000.

[18] S. Shakkottai and R. Srikant. Scheduling real-time traffic with deadlines over a wireless channel. *ACM/Baltzer Wireless Networks Journal*, 8(1):13–26, Jan. 2002.

[19] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, 1996.

[20] G. Song and Y. Li. Cross-layer optimization for OFDM wireless networks part I: Theoretical framework. *IEEE Transactions on Wireless Communications*, 4(2), Mar. 2005.

[21] D. Tse and S. Hanly. Multi-access fading channels: Part I: Polymatroid structure, optimal resource allocation and throughput capacities. *IEEE Transactions on Information Theory*, 44(7):2796–2815, Nov. 1998.

[22] P. Viswanath, D. Tse, and R. Laroia. Opportunistic beamforming using dumb antennas. *IEEE Transactions on Information Theory*, 48(6), June 2002.

[23] I. Wong, Z. Shen, J. Andrews, and B. L. Evans. A low complexity algorithm for proportional resource allocation in OFDMA systems. *IEEE International Signal Processing Systems Workshop*, Oct 2004.

PLACE PHOTO HERE

Reuven Cohen (M'93, SM'99) received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the Technion - Israel Institute of Technology, completing his Ph.D. studies in 1991. From 1991 to 1993, he was with the IBM T.J. Watson Research Center, working on protocols for high speed networks. Since 1993, he has been a professor in the Department of Computer Science at the Technion. He has also been a consultant for numerous companies, mainly in the context of protocols and architectures for broadband access networks. Dr. Cohen has served as an editor of the IEEE/ACM Transactions on Networking, and the ACM/Kluwer Journal on Wireless Networks (WINET). Dr. Cohen is a senior member of the IEEE and heads the Israeli chapter of the IEEE Communications Society.

PLACE PHOTO HERE

Liran Katzir Liran Katzir received his B.A., M.A., and Ph.D. degrees in computer science from the Technion-Israel Institute of Technology, Haifa, Israel, in 2001, 2005, and 2008 respectively. His PhD. thesis is about scheduling in advanced wireless networks.

Dr. Katzir is now a member of the networking research group in the Technion's CS department, working on technologies for the fourth generation cellular network.