Throughput-optimal Resource Allocation in LTE-Advanced with Distributed Antennas

Giovanni Accongiagioco
IMT Lucca
Lucca, Italy
giovanni.accongiagioco@imtlucca.it

Matteo Andreozzi
NVIDIA
Cambridge (UK)
mandreozzi@nvidia.com

Daniele Migliorini
Lepida SpA
Bologna, Italy
daniele.migliorini@lepida.it

Giovanni Stea
University of Pisa
Pisa, Italy
g.stea@iet.unipi.it

Abstract

Distributed antennas are envisaged for LTE-Advanced deployments in order to improve the coverage and increase the cell throughput. The latter in turn depends on how resources are allocated to the User Equipments (UEs) at the MAC layer. In this paper we discuss how to allocate resources to UEs so as to maximize the cell throughput, given that UEs may receive from several antennas simultaneously. We first show that the problem is both NP-hard and APX-hard, i.e. no polynomial-time algorithm exists that approximates the optimum within a constant factor. Hence, we propose and evaluate two polynomial-time heuristics whose complexity is feasible for practical purposes. Our simulative analysis shows that, in practical scenarios, the two heuristics are highly accurate.

Keywords: LTE, Distributed Antennas, Complexity, Spatial Multiplexing

1 Introduction

The Long Term Evolution (LTE) of the Universal Mobile Telecommunication System (UMTS) [1] is gaining progressive hold as an access network for Internet services, thanks to its foreseen near-ubiquitous coverage and high bandwidth. In such a system, a central base station or eNodeB shares radio resources among a number of User Equipments (UEs), i.e. handheld devices, laptops or home gateways, using Orthogonal Frequency Division Multiplexing Access (OFDMA) in the downlink. On each Transmission Time Interval (TTI), the eNodeB allocates a *frame*, which contains a finite number of resource blocks (RBs) to be shared among the UEs. A RB may carry a variable number of bits, depending on the modulation used to encode the RB, which is suggested by the Channel Quality Indicator (CQI) advertised by the UE.

The new standard of LTE, called LTE-Advanced (LTE-A) [2], allows several architectural enhancements. One of the most prominent is the *Distributed Antenna System* (DAS) deployment, in which the eNodeB acts as a hub for a number of physically distributed antennas (or *Remote Units*, *RU*). RUs are linked to the eNodeB via high-speed wired connections, and they transmit one frame each. Resource allocation is however centralized, and decided by the eNodeB on each TTI. A UE may experience different channel quality with respect to different RUs. Using a *spatial multiplexing* transmission mode, a UE may receive independent transmissions from several RUs simultaneously on the same RB, which it can decode relying on *spatial separation*. Accordingly, in a DAS deployment, maximizing the cell throughput implies deciding not only *to whom* RBs should be given (e.g., to the UE with the highest CQI), but also *through which RU(s)* among those that can target a single UE at a given time.

This paper tackles the problem of resource allocation at the MAC level in a LTE-A cell using DAS, in the downlink direction. The objective is to allocate RBs to UEs so as to maximize the overall cell throughput. We first prove that a maximum-throughput allocation can be found in polynomial time if one assumes that the eNodeB has an *infinite* supply of data for each UE (*full buffer* assumption). That assumption, however, is unrealistic, since many applications have periodic (e.g., voice and video) or bursty (e.g., web) arrival patterns. Unfortunately, when finite buffers are considered, the problem of maximum-throughput allocation becomes NP-hard. Therefore, we propose polynomial-time heuristics that approximate the optimal solution. We evaluate their performance in the two dimensions of complexity and accuracy, under several simulation scenarios. Our results show that the heuristics are near-optimal in practical deployments.

To the best of our knowledge, this is the first work that deals with these specific problem and settings. The problem of MAC-level resource allocation on LTE-A with DAS is relatively new, which justifies the paucity of the literature on the subject. Some related work exists on OFDM networks (e.g., [6]-[8]), where each antenna can serve a *single* UE. Under this hypothesis the problem can be solved optimally in polynomial time. However, in a LTE-A network a frame carries up to ten Kb hence allowing for multiple UEs in a single frame is essential. The optimal solutions computed for OFDM systems are therefore useless in practice.

The rest of the paper is organized as follows: Section 2 provides background on the relevant features of the LTE technology. Section 3 describes the system model and the hardness results. Our heuristics are presented in Section 4, and evaluated in Section 5. Section 6 reviews the related work, and conclusions are reported in Section 7.

2 LTE-Advanced with Distributed Antennas

In this section we describe the features of the LTE-A system that are more relevant to the downlink resource allocation problem at the MAC layer. Distributed Antenna Systems (DAS) can be employed in LTE-A to increase coverage and/or transmission rate. In a DAS deployment, the eNodeB is connected via a fiber interface to a set of M spatially distributed antennas or *Remote Units* (RU), whose coverage may overlap, as shown in Fig. 1. Transmissions are arranged in time slots called Transmission Time Intervals, (TTIs), whose duration is 1ms. The eNodeB schedules transmissions by composing M *frames* and transmitting each of them through one RU on each TTI. At the *logical* (MAC) level a *frame*, is a vector of (*Virtual*) *Resource Blocks* (RBs), each one of which is transmitted to one UE only (unless multi-user MIMO is employed, which is outside the scope of this paper) on a different frequency. Each RB carries a fixed number of symbols, which translate to different amounts of bits depending on the modulation used on that RB. In general, more information-dense modulations (e.g., 64QAM, yielding up to 6 useful bits per symbol) are favored when a better channel to the UE is perceived. The quality of the wireless channel varies over time and is generally different from one UE to the other. For this reason, UEs report their perceived downlink channel states to the eNodeB scheduler as Channel Quality Indicators (CQIs) to the RUs the UE is associated to. The CQI is an index in a standard table, computed by the UE according to the measured Signal to Noise and Interference Ratio (SNRI), and influences the Transmission Block Size (TBS) that the eNodeB should use, i.e., the number of bits per RB. The UE may either report *one* CQI (called *wideband* CQI), describing the average channel conditions over the whole spectrum, or several *per sub-band* CQIs, each one describing the channel conditions in a fraction of the spectrum.

A single UE may also receive transmissions from different RUs on the same time-frequency resource, exploiting a feature known as *Spatial Multiplexing*[1]. Those transmissions are in fact *spatially separated*, and can therefore be decoded at the UE with a sufficiently high probability. Spatial separation can be assumed when the distance between RUs (which is in the order of meters, or tens and hundreds thereof) is larger than half the wavelength of the LTE transmission, which is in the order of 15cm. This hypothesis is verified in practice. Furthermore, geographically remote RUs (e.g., deployed around the perimeter of a cell) can be selectively activated, and – when active – can target a subset of UEs (e.g., thanks to beamforming techniques, as allowed by TM 8 and 9, [20], [21]). In the current standard, UEs report CQIs for at most two codewords (i.e., independent spatial streams). Furthermore, the accuracy of CQI reporting depends on the UE reporting scheme. Depending on whether the reporting is periodic or aperiodic, the CQI for the second codeword may be reported fully or differentially with respect to the first one (on four bits), hence with reduced accuracy.

In addition to the CQIs, a UE reports its *Rank Indicator* (RI) to the eNodeB. This number is the rank of the receiver channel matrix, and defines the maximum number of *spatial layers* that the UE can decode, as a recommendation to the eNodeB. This number, in turn, bounds from the above the number of different spatial streams that the UE may receive. The RI changes over time, depending on the channel conditions.

[1] Spatial Multiplexing is allowed in several transmission modes (TM) of the current LTE-A standard. The ones of interest for this paper are TM 8 and 9.

UE traffic is physically buffered at the eNodeB. In order to build a schedule in a TTI, the eNodeB scheduler selects *which UEs* are going to be targeted, through *which and how many RU(s)*, using *which TBS* to guarantee reliable transmission, and employing *how many RBs*. These decisions are made, either sequentially or jointly, based on the reported CQIs, the RIs, the backlog and type of traffic of each UE, the QoS requirements, etc. The goal is to maximize the cell throughput. Of course, a schedule consisting of M frames has to be built in 1ms time, which limits the complexity of the decisions.

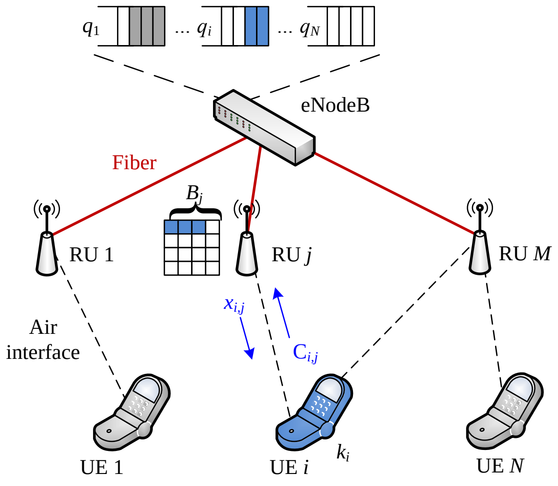

Fig. 1 – A DAS deployment

3 System Model and Hardness results

We focus on the downlink of an LTE-A cell, which is managed, at the MAC layer, by an eNodeB scheduling entity.

We focus on a single cell, whose eNodeB scheduler coordinates M distributed RUs, hence builds M frames. Each RU j transmits a frame of B_j RBs (it is not uncommon to have $B_j = B, \quad \forall j$). The cell provides service for N UEs, and we denote with q_i the backlog (physically queued at the eNodeB) destined to UE i. With a little abuse of terminology, we will henceforth concisely define the latter "UE backlog", and accordingly we will call *idle* those UEs whose backlog is null. While we make no specific hypothesis on how the RUs are deployed, it is possible that each of them is only heard by a subset of UEs, and that a single UE does not hear all the RUs simultaneously, especially if they are deployed with partial overlapping coverage. Furthermore, we assume that RUs can selectively target some of the UEs within their coverage area, e.g. through beamforming. We assume that the eNodeB possesses the following information:

- An estimate $C_{i,j}$ of the number of bits that can be transmitted in an RB by RU j to UE i (assumed to be the same for all RBs, hence coherent with wideband CQI reporting). We call $C_{i,j}$ the *capacity*, which may differ, for the same UE, from one RU to the other, e.g., because of different pathloss.
- An estimate of the maximum number of simultaneous spatial streams that UE i may receive, called k_i. If only SISO transmissions are assumed, then it is $k_i = 1$ for all UEs.

The exact means through which these values are obtained by the eNodeB are outside the scope of this paper. Some of these are obviously obtained from UE reports (e.g., the capacity from the CQI and the number of spatial streams from the RI). However, in practice UE reports are not necessarily complete, accurate, or available on every TTI, and can be supplemented through other means, such as proprietary physical layer measurements, data acquired at previous TTIs, default values, etc.

We denote with $K = \sum_{i=1}^{N} k_i$ the upper bound on the total number of receivable spatial streams. We denote with $x_{i,j}$ the number of RBs allocated by RU j to UE i. Thus, $C_{i,j} \cdot x_{i,j}$ is the overall number of bits that UE i receives from RU j. Our objective is to compute, on each TTI, the solution to the *Throughput-optimal Resource Allocation Problem (TORAP)*. Given the above information, we can formulate it as follows:

$$\begin{aligned}
&\max \sum_{i=1}^{N} \sum_{j=1}^{M} \left(C_{i,j} \cdot x_{i,j} - p_{i,j} \right) && && \\
&s.t. \quad \sum_{j=1}^{M} \left(C_{i,j} \cdot x_{i,j} - p_{i,j} \right) \le q_i && \forall i && (i) \\
&\qquad p_{i,j} \le C_{i,j} - 1 && \forall i,j && (ii) \\
&\qquad \sum_{i=1}^{N} x_{i,j} \le B_j && \forall j && (iii) \qquad (1)\\
&\qquad \sum_{j=1}^{M} b_{i,j} \le k_i && \forall i && (iv) \\
&\qquad x_{i,j} \ge b_{i,j} && \forall i,j && (v) \\
&\qquad x_{i,j} \le b_{i,j} \cdot B && \forall i,j && (vi) \\
&\qquad b_{i,j} \in [0,1], \quad p_{i,j}, x_{i,j} \in Z^{+} && \forall i,j && (vii)
\end{aligned}$$

The following modeling variables are introduced:

- $p_{i,j}$ are the *padding* bits transmitted by antenna j to UE i. Since RBs carry a *fixed* number of bits, partially occupied RBs have to be padded. Obviously, padding bits cannot be counted in the objective function, otherwise an optimal solution might include RBs allocated to UEs with few useful bits and high capacities.

- $b_{i,j}$ is a binary variable stating whether RU j is actually serving UE i (i.e., eating away one of its spatial streams).

The objective function is the sum of *useful* bits transmitted by all RUs to all UEs, i.e., minus the padding. Constraint (*i*) ensures that each UE receives no more RBs than those needed to clear its backlog, whereas constraint (*ii*) bounds the padding from above, preventing the scheduler to allocate RBs entirely consisting of padding. Constraint (*iii*) enforces the limit on the number of RBs in each RU's frame. Constraint (*iv*) limits the number of simultaneously received spatial streams to each UE's maximum, whereas constraints (*v*-*vi*) force $x_{i,j}$ to be strictly positive if $b_{i,j}=1$, and null otherwise. B is a constant such that $B \geq B_j \ \forall j$, so that (*vi*) is inactive when $b_{i,j}=1$.

Problem (1) is a Mixed Integer-Linear Problem (*MILP*), with $O(N \cdot M)$ variables and $O(N \cdot M)$ constraints. MILPs are known to be NP-hard in general [13]. Hereafter we discuss hardness in more detail. At the end of this section, we discuss some generalizations of the models and results.

3.1 Hardness results

Hereafter, we prove theorems that define the hardness of the TORAP problem. We first show that polynomial-time algorithms exist for the *full-buffer* case, i.e. when all backlogs q_i are so large that each UE can be allocated all the available frame space at every RU. Then, we show that the problem is instead NP-hard in the *finite-buffer* case, i.e., when constraint (*i*) in (1) can actually be active.

Theorem 1: *Under the full-buffer hypothesis, i.e. if:*

$$\forall i,\ q_i \geq \sum_{j=1}^{M} C_{i,j} \cdot B_j \ , \qquad (2)$$

the optimal solution to the TORAP problem can be found in $O(K^3)$ operations.

Proof: Inequality (2) ensures that each UE's backlog is large enough to fill every frame, which implies that no padding is required, hence $p_{i,j}=0 \ \forall i,j$. Furthermore, under (2), it is not restrictive to assume that each RU j serves only *one* UE. In fact, if the optimal solution is such that *both* UEs i and h are served by j, then it can only be that $C_{i,j}=C_{h,j}$, i.e. a solution where j serves only either of them is optimal as well (both i and h's buffers being full enough). Hence the problem is to assign *one* UE to each RU, subject to the stream constraint (*iv*). Assume for simplicity that $k_i=1 \, \forall i$ (we will remove this assumption later on in the proof). Then, the problem is equivalent to an *assignment problem* (AP) on a bipartite graph whose nodes are the N UEs and the M RUs respectively, and whose arcs are labeled with $C_{i,j} \cdot B_j$, i.e. the number of bits that frame j would carry if given to UE i. An AP can be solved in polynomial time via the *Hungarian Algorithm* (*HA*) [14]. HA matches elements of two *idempotent* sets X and Y so as to *minimize the total cost*, assuming that each match has a non-negative cost $c_{x,y}$. In our case, the two sets include N UEs and M RUs, with $N > M$ in practical cases. The mismatch between the two cardinalities can be circumvented by adding $N-M$ *dummy* RUs. The matching costs can be computed as:

$$c_{i,j} = \begin{cases} C_{\max} - C_{i,j} & j \leq M \\ C_{\max} & otherwise \end{cases}, \qquad (3)$$

where $C_{\max}$ is the maximum number of bits per RB allowed by the standard. This way a higher capacity implies a lower cost, and dummy RUs (i.e., those with $j > M$) are never chosen except as a last resort.

The complexity of HA is the cube of the sets' cardinality, hence $O(N^3)$. The result is a set of couples $S=\{(i,j) \mid 1 \leq i,j \leq N\}$ describing the minimum-cost match, from which the relevant subset $S' = \{(i,j) \in S \mid j \leq M\}$, which only includes matches between UEs and real RUs, can be extracted in linear time.

Assume now that $k_i \geq 1 \, \forall i$. Then the problem can still be solved using HA. In fact, each UE can be selected for a match with a RU k_i independently. Thus, it is enough that we replace each UE i with k_i identical copies of itself (thus obtaining a set of K spatial streams), add $K-M$ dummy RUs to match the cardinality of the set of spatial streams, and compute the costs according to the obvious generalization of (3). Under these settings, HA computes the optimal match in $O(K^3)$ operations.

□

Author's version of:
G. Accongiagioco, M. Andreozzi, D. Migliorini, G. Stea, "Throughput-optimal Resource Allocation in LTE-Advanced with Distributed Antennas" Elsevier Computer Networks, vol. 57(2013), pp. 3997-4009 (December 2013), DOI 10.1016/j.comnet.2013.10.002

Still under the full-buffer hypothesis (2), an even simpler solution exists if every UE can receive simultaneously from *all* the M RUs, as shown by the following corollary.

Corollary 2: *Under the full-buffer hypothesis* (2), *if also*

$$k_i \geq M \quad \forall i \tag{4}$$

then the optimal solution to the TORAP problem can be found in $O(N \cdot M)$ *operations.*

Proof: Under (4), the stream constraint (*iv*) is never active, and the problem can be solved as follows: iterate on all the M RUs: at each RU j, pick the highest capacity $C_{i,j}$ and allocate the whole frame to UE i. This is obviously a safe decision, since UE i will still be able to exploit every other frame at subsequent iterations. The whole cycle completes in $O(N \cdot M)$ operations.

□

Although it may be regarded as useful to compute capacity bounds, the full-buffer hypothesis is unrealistic when dealing with real-life applications. VoIP traffic, for instance, has periodic arrivals, with large periods (e.g., 20 TTI), and small packets (e.g., 32 bytes). Compressed real-time video is sent periodically with large inter-frame intervals (e.g., 40 TTI), and variable-sized packets. Finally, web-browsing traffic is intermittent. Unfortunately, under a more realistic *finite-buffer* hypothesis, the TORAP problem becomes NP-hard, as proved by the following theorem:

Theorem 3: *With finite buffers, the TORAP problem is NP-hard.*

Proof: Consider the following *simplified* version of the TORAP problem: $B_j = 1 \ \ \forall j$. In this case, $x_{i,j} \in \{0,1\}$, which makes variables $b_{i,j}$ unnecessary. Assume also that $k_i \geq M \ \ \forall i$, so that constraint (*iv*) is not necessary. Furthermore, let us assume that padding can be counted as useful transmission. We show that computing an optimal assignment of $x_{i,j}$ under these hypotheses is NP-hard. The problem can be rewritten as follows:

$$\begin{aligned} &\max \sum\nolimits_{i=1}^{N} \sum\nolimits_{j=1}^{M} C_{i,j} \cdot x_{i,j} \\ &s.t. \\ &\sum\nolimits_{j=1}^{M} C_{i,j} \cdot x_{i,j} \leq q_i \qquad \forall i \qquad (i) \\ &\sum\nolimits_{i=1}^{N} x_{i,j} \leq 1 \qquad \forall j \qquad (ii) \\ &x_{i,j} \in [0,1] \qquad \forall i,j \quad (iii) \end{aligned} \tag{5}$$

Problem (5) is a *Generalized Assignment Problem* (*GAP*) [15], where N *agents* (the UEs) are associated to M *tasks* (the RUs). Each task can be solved by one agent only, although each agent can solve more than one task. The $C_{i,j}$ incidentally represent both the unitary profits (in the objective function) and the costs (within constraint *(i)*), and q_i is the overall agent budget. The GAP problem is NP-hard [15].

□

Now, if a *simplified* version of the TORAP is NP-hard, the TORAP cannot be any easier. Interestingly enough, the GAP is also *APX-hard*, meaning that no polynomial-time α-approximation algorithm exists for the latter for any α [16]. In other words, heuristic solutions computed in polynomial time may be arbitrarily distant from the optimum in a worst case. We cannot afford non-polynomial heuristics, since the number of UEs is easily in the hundreds and allocations are to be carried out within 1ms. Thus, we aim our search towards *simple* polynomial-time heuristics, fast enough to build a schedule in a TTI. Before doing that, we discuss some implications of the hardness results exposed so far.

3.2 Discussion and generalizations

First of all, we observe that the hardness result of Theorem 3 is indeed more general. Consider for instance a different resource allocation problem, whose objective is to achieve a *proportional fair (PF)* allocation. Under PF, UEs are ranked by decreasing *PF score*. The latter measures the *relative* channel condition with respect to the recent history of the same UE. The PF score is equal to $C_{i,j}/R_i$, R_i being a constant that represents the historical rate, usually obtained as an exponential average of the rates measured at the previous TTIs. Furthermore, the PF score with multiple antennas is the sum of the PF scores of single antennas. One can easily see that problem (5) does not change its structure (hence its hardness) if we substitute $C_{i,j}/R_i$ for $C_{i,j}$ as a unitary profit in the objective function. Within this paper, we will focus on the maximum throughput objective, and we will devise heuristics that work well for that objective. It is highly likely that the same heuristics can be adapted, with little to no tuning, to the PF case as well.

Furthermore, the models and algorithms presented in this paper, devised for the *downlink* direction, can be straightforwardly mapped to the *uplink* direction as well. In the uplink, the eNodeB knows an estimate of the backlog of the UEs via periodic and/or solicited *Buffer Status Reports (BSRs)* transmitted by the UEs. Instead of scheduling traffic, the eNodeB schedules uplink *grants*, telling each UE which RBs to use in the subsequent uplink frame. The uplink equivalents of CQIs are estimated by the eNodeB via a *sounding* procedure [3]. Finally, spatial multiplexing techniques can also be used in the uplink. Therefore, the same information is available at the eNodeB to formulate an uplink equivalent of the TORAP, although that information is obtained through different means. Hence, the algorithms that we present in the next section can be adapted to the uplink direction as well, *mutatis mutandis*.

4 Polynomial-time Heuristics

Having shown that the TORAP problem is NP-hard and APX-hard, we now present two greedy polynomial-time heuristics that exhibit reasonably small complexity. In Section 5 we will evaluate their accuracy in several scenarios.

4.1 Algorithm 1

With reference to the pseudo-code of Fig. 2, Algorithm 1 works as follows: we associate to each (UE, RU) pair an *allowance* $A_{i,j}$, i.e. the number of *useful* bits that fit in a single RB. Initially (function `Init`), it is $A_{i,j} = \min\{C_{i,j}, q_i\}$, for all the pairs worth considering in the allocation. The algorithm (function `Algorithm1`) cyclically selects the largest allowance, be it $A_{i,j}$ (line 12). For the (i,j) (UE, RU) pair, the minimum between $\lfloor q_i / A_{i,j} \rfloor$ (i.e. all the RBs that UE *i* can receive from RU *j* without resorting to padding), and the number of available RBs B_j is computed (line 14) and allocated (line 15). Furthermore, UE *i*'s backlog is adjusted to reflect the allocation (line 16) and the number of available RBs for RU *j* is modified accordingly (line 17). After this allocation (which always gives away at least one RB), if RU *j* has no leftover RBs, its allowances are zeroed (line 18), so that it cannot be selected again. When UE *i* is matched to a new RU, k_i is decreased (line 13), and when k_i reaches zero, all the $A_{i,j}$ for which $x_{i,j} = 0$ (i.e., those related to RUs which have not been matched to *i* yet) are zeroed for consistency (line 20). Finally, when the UE buffer is not enough to fill one RB, i.e. $q_i < A_{i,j}$ (including when $q_i = 0$), we need to clip all the allowances of UE *i* to q_i (line 19), lest we count padding as useful transmissions (or, worse yet, keep considering idle UEs for allocation). The cycle ends when all allowances are null (line 11). When that happens, variables $x_{i,j}$ describe the final allocation. We show an execution of Algorithm 1 in a toy example.

Example 1

We start from the data in Table 1, obtained for a system with $N = 3$, $M = 3$, $k_i = 2\ \forall i$, $B_j = 3\ \forall j$. Each UE has enough backlog to fill up at least one RB of each RU, hence $A_{i,j} = C_{i,j}$. The maximum allowance is $A_{3,2} = 10$, hence $(3,2)$ is the first match. UE 3 may receive two spatial streams, and it has enough backlog to fill two RBs completely. Hence, two RBs from RU 2 are given away. UE 3 is left with 5 bytes worth of backlog. Hence $A_{3,2} \leftarrow 5$, all the allowances for UE 3 are capped to 5, and $k_3 \leftarrow 1$. The new configuration is reported in Table 2.1. The maximum allowance is now $A_{1,2} = A_{2,1} = 9$. We break the tie (arbitrarily) by selecting $(1,2)$, and allocate one RB to UE 1. RU 2 is no more eligible; hence its column is zeroed as shown in Table 2.2. The algorithm now picks $A_{2,1} = 9$ as the maximum and allocates two RBs from RU 1 to UE 2. The allocation occupies one spatial stream from UE 2, and the new allowances for UE 2 are capped to the residual backlog of 2 bytes, as shown in Table 2.3. As a fourth step, we pick $A_{1,1} = 8$, allocate the leftover RB at RU 1 to UE 1, and zero both column 1 *and* row 1, since UE 1 cannot receive any more spatial streams and a match with RU 3 is thus impossible. This is shown in Table 2.4. The three remaining RBs at RU 3 are assigned (in three subsequent iterations) as follows: first, *one* goes at UE 3 (with no padding), and then the remaining two are given to UE 2 and UE 3 (in any order, given the tie), suitably padded. The final status, reported in Table 2.5, shows that 13 bytes remain buffered at UE 1.

```
1.  Function Init
2.  ∀i,j
3.      if $B_j>0$ and $q_i>0$ and $k_i>0$ then
4.        $A_{i,j} = \min\{q_i, C_{i,j}\}$
5.      else
6.        $A_{i,j} = 0$
7.  end function
8.
9.  Function Algorithm1
10.   Init
11.   while $\max_{ij}\{A_{i,j}\}>0$ do
12.     $(i,j) = \arg\max_{ij}\{A_{i,j}\}$
13.     if $x_{i,j}=0$ then $k_i=k_i-1$
14.     $b=\min\{\lfloor q_i/A_{i,j}\rfloor, B_j\}$
15.     $x_{i,j} = x_{i,j} + b$
16.     $q_i = q_i - b\times A_{i,j}$
17.     $B_j = B_j - b$
18.     if $B_j = 0$ then $\forall i$, $A_{i,j}=0$
19.     if $q_i < A_{i,j}$ then $\forall j$, $A_{i,j}=\min\{A_{i,j}, q_i\}$
20.     if $k_i = 0$ then $\forall j \mid x_{i,j}=0$, $A_{i,j}=0$
21. end function
```

Fig. 2 – Pseudo-code for Algorithm 1.

Table 1 – Initial data for the example

$C_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	8	9	7	**2**	**30**
UE2	9	2	6	**2**	**20**
UE3	3	10	3	**2**	**25**
B_j	**3**	**3**	**3**		

Table 2 – Data for each iteration in Algorithm 1

1.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	8	9	7	**2**	**30**
UE2	9	2	6	**2**	**20**
UE3	3	5	3	**1**	**5**
B_j	**3**	**1**	**3**		

2.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	8	0	7	**1**	**21**
UE2	9	0	6	**2**	**20**
UE3	3	0	3	**1**	**5**
B_j	**3**	**0**	**3**		

3.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	8	0	7	**1**	**21**
UE2	2	0	2	**1**	**2**
UE3	3	0	3	**1**	**5**
B_j	**1**	**0**	**3**		

4.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	0	0	0	**0**	**13**
UE2	0	0	2	**1**	**2**
UE3	0	0	3	**1**	**5**
B_j	**0**	**0**	**3**		

5.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	0	0	0	**0**	**13**
UE2	0	0	0	**0**	**0**
UE3	0	0	0	**0**	**0**
B_j	**0**	**0**	**0**		

We now compute the complexity of Algorithm 1:

Property 4: *The complexity of Algorithm 1 is*

$$\min\left\{O\left(N^2\cdot M^2\right),O\left(N\cdot M^2\cdot B\right)\right\}.$$

Proof: the `Init` function takes $O(N\cdot M)$ operations. As far as the `Algorithm1` function is concerned, the number of iterations in the `while` cycle is bounded by both $O(N\cdot M)$ and $O(M\cdot B)$. In fact, on each iteration, at least one of the conditions of the `if`s stated in lines 18-20 becomes true. Thus the same pair (i,j) can be considered at most *twice* in the whole cycle before becoming null: a first time to allocate one or more RBs without padding, and possibly a second time to allocate *one* RB with padding. For the same reason, on each iteration at least one RB is allocated, hence the number of iteration is also bounded by the overall number of available RBs, i.e. $O(M\cdot B)$. Within each iteration, searching for the max in the allowance matrix takes $O(N\cdot M)$ operations. Modifying the allowances as specified in the `if`s of lines 18-20 takes either $O(N)$ or $O(M)$ operations. Thus, the cost per iteration is $O(N\cdot M)$, hence the thesis.

□

Since sorting is required to find the maximum allowance, the alert reader may wonder whether a sorted-list implementation may prove more efficient. The answer is the following:

Property 5: *An equivalent implementation of Algorithm 1 that uses a max heap has a complexity equal to:*

$$\min\left\{\begin{array}{l}O\left(N\cdot M^2\cdot\log\left(N\cdot M\right)\right),\\ O\left(M\cdot\left(M\cdot B+N\right)\cdot\log\left(N\cdot M\right)\right)\end{array}\right\}$$

The proof of Property 5 is reported in the Appendix, and it is based on rewriting Algorithm 1 employing i) a max heap to sort the allowances in decreasing order, and ii) circular lists to link all the allowances related to the same UE and RU.

□

The above results show that Algorithm 1 scales *linearly* with the number of UEs and the system bandwidth (i.e., the number of RBs). Algorithm 1 is thus scalable enough to be implemented on real equipment, where up to few hundreds UE have to be served by less than ten RU, each with up to a hundred RBs, and schedules must be computed within 1ms.

4.2 Algorithm 2

We now describe a different algorithm, which uses the Hungarian Algorithm (HA) as a subroutine. The interested reader is referred to [14] for a detailed description of the HA. For our purposes, it is sufficient to recall that it takes as an input a *square* cost matrix, and it outputs the minimum-cost match between each row and column in the cost matrix. In our case, we need to match *spatial streams* to *RUs*. Therefore, assuming that $K \gg M$, we must prepare an input consisting of a $K \times K$ *expanded cost matrix*. In the pseudo-code of Fig. 3, this is done within function `HA`. The latter starts from the rectangular $N \times M$ allowance matrix, and computes costs $c_{i,j}$ as the complement to the maximum capacity of these allowances (line 2). It then computes the expanded cost matrix by: i) replicating $k_i - 1$ times each row of costs, (lines 3-9), also keeping track of which row refers to which UE via the *UE(i)* map, and ii) adding $K - M$ dummy RUs, i.e. $K - M$ columns of maximum costs (line 10). The function then runs the HA on the expanded cost matrix (line 11), so as to match each *spatial stream* with a RU, and returns the set of matches, purged of those pertaining to dummy RUs or non-eligible UEs/RUs (line 12), sorted by increasing RU index.

The structure of function `Algorithm2` is similar to that of `Algorithm1`. The main difference is that the former, thanks to the HA subroutine, computes *many* matches at the same iteration. More specifically, it computes the initial allowances $A_{i,j}$, using the same `Init` function as Algorithm 1, then runs the HA to compute a set of matches I (line 18). For each match in I, it allocates as many RBs as possible (lines 23-27) without resorting to padding. If a UE/RU is not eligible anymore, or the k_i constraint prevents any more matches (lines 28-30) the UE/RU is removed from the allocation by zeroing the related allowances. Note that, since set I may contain more than one pair for the same UE, it is possible that a pair $(i,j) \in I$ is considered when $q_i < A_{i,j}$. In this case, the pair is skipped (lines 20-22). The cycle iterates while there are positive allowances.

We show how Algorithm 2 works on the example used for Algorithm 1.

Example 2

Starting from Table 1, HA returns the following match: $\{(1,3),(2,1),(3,2)\}$. UE 1 transmits 3 RBs (thus depleting RU 3), UE 2 transmits 2 RB, and UE 3 transmits 2 RBs. The allocation is reported in Table 3.1. A new iteration is required, and the outcome of HA is $\{(1,1),(3,2)\}$. UEs 1 and 2 can only transmit one RB, thus depleting the two remaining RUs. The 2nd iteration is reported in Table 3.2. Algorithm 2 scores higher than Algorithm 1 in this example (three bytes remaining in the UE queues against 13), using fewer spatial streams (four against six).

As far as complexity is concerned, we can state the following

Property 6: *Algorithm 2 has* $O(M \cdot B \cdot K^3)$ *complexity.*

Proof: function `HA` has $O(K^3)$ complexity [14], and computes $O(M)$ matches. The cycles at lines 28-30 may happen only once in the execution of an algorithm, hence their total cost is $O(N)$ and $O(M)$ respectively. At each iteration, at least one RB is allocated, hence the thesis. □

Algorithm 2 is thus significantly more complex than Algorithm 1, as it scales with the cube of the number of streams (hence, of UEs). We remark that, under the full-buffer hypothesis, it is also *optimal*, as discussed in Section 3.

```
Function HA (input: NxM allowance matrix)
   $\forall i,j$, $c_{i,j} = C_{max} - A_{i,j}$
   for i=1 to N UE(i)=i
   set rowcount=N
   for each $i \in$ UEs | $k_i > 1$
     for x = 1 to $k_i - 1$
     rowcount = rowcount+1
     UE(rowcount)=i
     copy row i to row rowcount
   add K-M columns of max costs to C
   S = execute HA on matrix C
   return S'={(UE(i),j)| (i,j)$\in$S and $A_{i,j} > 0$}
end function

Function Algorithm2
   Init
   while $\max_{ij}\{A_{i,j}\} > 0$ do
     I=HA(allowances)
     for all (i,j) in I
       if $q_i < A_{i,j}$ then
         set $A_{i,j} = q_i$
         continue for
       if $x_{i,j} = 0$ then $k_i = k_i - 1$
       b=min {floor ($q_i / A_{i,j}$), $B_j$}
       $x_{i,j} = x_{i,j} + b$
       $q_i = q_i - b \times A_{i,j}$
       $B_j = B_j - b$
       if $B_j = 0$ then $\forall i$, $A_{i,j} = 0$
       if $q_i = 0$ then $\forall j$, $A_{i,j} = 0$
       if $k_i = 0$ then $\forall j \mid x_{i,j} = 0$, $A_{i,j} = 0$
end function
```

Fig. 3 – Pseudo-code for Algorithm 2

Table 3 – Data for the iterations in Algorithm 2

1.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	8	9	0	**1**	**9**
UE2	2	2	0	**1**	**2**
UE3	3	5	0	**1**	**5**
B_j	**1**	**1**	**0**		

2.

$A_{i,j}$	**RU 1**	**RU 2**	**RU 3**	k_i	q_i
UE1	0	0	0	**0**	**1**
UE2	0	0	0	**1**	**2**
UE3	0	0	0	**1**	**0**
B_j	**0**	**0**	**0**		

5 Performance Evaluation

In this section, we evaluate the accuracy of Algorithm 1 and 2, i.e. their distance to the optimum. The evaluation is carried out via simulation. The simulator, written in PHP and Python, uses CPLEX [18] to solve optimization problems. It can run in both *snapshot* mode, to compare the outcome of an execution of the heuristics to the optimum, and *time-based* mode, by feeding back the output of each snapshot as input to the subsequent snapshot.

The algorithms are initially evaluated in ideal conditions. We assume fluid traffic and no data corruption. Furthermore, we assume that the UEs report fresh CQI values on every TTI, for each (UE, RU) couple. This allows us to have meaningful $C_{i,j}$ values for every couple (*i*,*j*). The above ideal scenario allows us to explore all the range of possibilities in the search for optimal solutions. Later on, we show that non-ideal reporting hardly affects the performance metrics at all. The CQI values reported by the UEs are shown in Table 4.

Table 4 – Capacities associated to CQI indexes.

CQI Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Capacities (Bytes/RB)	0	3	3	6	11	15	20	25	36	39	50	63	72	80	93	93

UEs may report their CQIs according to either:

- a *uniform* distribution, which assumes no specific UE preference for a RU. In this case, CQI indexes are extracted uniformly in the interval $[1;15]$;
- a *dominant-RU* distribution. In this case, the CQIs indexes are partitioned into $L = [1;10]$ and $H = [11;15]$. The CQI for the dominant RU is chosen uniformly in H, and the ones for non-dominant RUs are selected uniformly in L.

In time-based simulations, the next CQI is computed by adding a uniform random integer in $[-2;+2]$ to the current one.

5.1 Snapshot simulations

We define the *optimality ratio R* of an algorithm as its throughput normalized to the optimum. We first investigate the impact of the various parameters (i.e., number of RUs, UEs, spatial streams, RBs, and offered load) on *R*.

1) Uniform-CQI scenario

We start with a symmetric case, with $N = 20$, $M = 6$, $B_j = 50 \quad \forall j$, $k_i = 2 \quad \forall i$, and $q_i = OL/N$, OL being the variable offered load. Fig. 4 shows the cell throughput (*CT*) as a function of OL for the *optimum* (left *y* axis), and the value of R for Algorithm 1 and 2 (right *y* axis), as an average of 20 replicas. Confidence intervals are small, and they are omitted for the sake of readability. The non-monotonic behavior of R can be explained as follows: when the offered load is low, the cell is lightly loaded and there is enough space for every UE, hence any strategy will yield maximum throughput. When the cell is saturated, we approach a full-buffer condition, where Algorithm 2 is optimal. In the middle zone, i.e., around the knee in the CT curve, the suboptimality of greedy decisions in both algorithms is more evident. Call $R_{\min}$ the minimum point in the R curve. To determine the accuracy of our algorithms, we study how sensitive $R_{\min}$ is to the system parameters (figures 5 to 8).

Fig. 5 shows that the number of RBs B_j is irrelevant, unless it is so small as to polarize the allocation. Fig. 6 shows how $R_{\min}$ varies with the number of spatial streams k_i (assuming $M = 10$ for this test to allow a higher gamut). Interestingly, both algorithms have the same performance *except* with single-stream UEs. In that case, Algorithm 2 performs considerably better. This is due to the fact that Algorithm 2 exploits a single iteration of the HA, which computes matches globally, whereas Algorithm 1 computes them iteratively. We then vary the number of UEs, spreading the same offered load among a different number of UEs. Fig. 7 shows that both algorithms fare better as the number of UEs grows, which is expectable since a higher multi-user diversity is known to improve allocations. The worst performance is achieved when the number of UEs is comparable to the number of RUs. Finally, we vary the number of RUs M in Fig. 8. The test shows that $R_{\min}$ decreases with M, confirming the above intuition. Note, however, that M is expected not to exceed few units.

Summing up, in a configuration with uniform CQIs, the two algorithms do not exhibit significant performance discrepancies, except for limit cases (e.g., $k_i = 1 \quad \forall i$), and $R_{\min}$ is above 90% in all cases of practical significance.

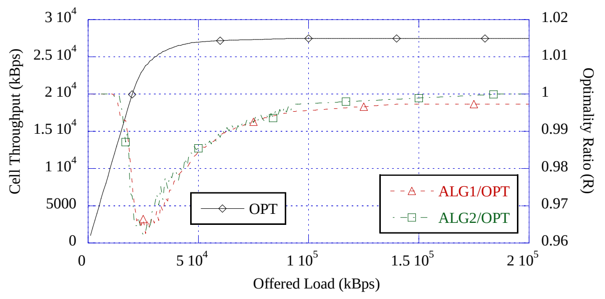

Fig. 4 – Cell throughput (left *y* axis) and optimality ratio (right *y* axis) vs. the offered load with uniform CQIs.

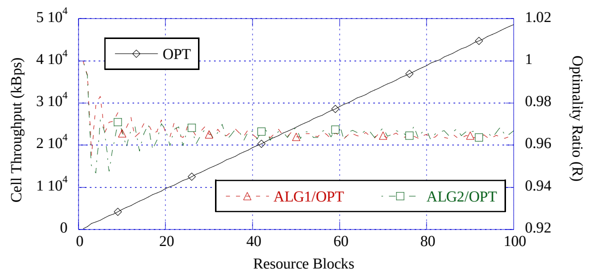

Fig. 5 – Cell throughput (left *y* axis) and $R_{\min}$ (right *y* axis) vs. number of RBs per RU.

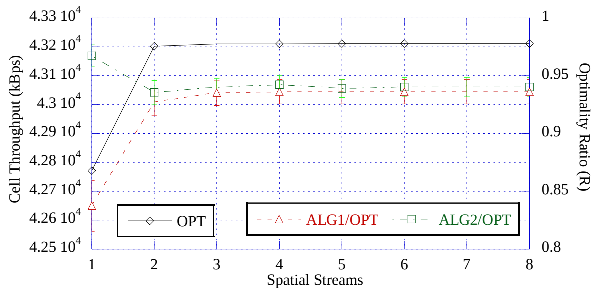

Fig. 6 – Cell throughput (left *y* axis) and $R_{\min}$ (right *y* axis) vs. number of spatial streams per UE.

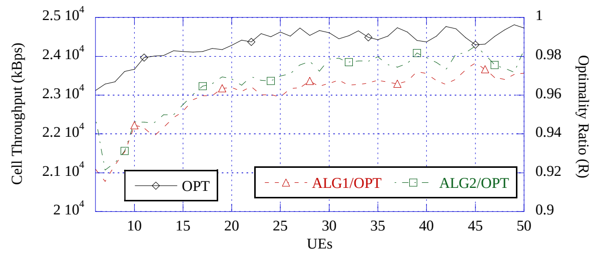

Fig. 7 – Cell throughput (left *y* axis) and $R_{\min}$ (right *y* axis) vs. number of UEs.

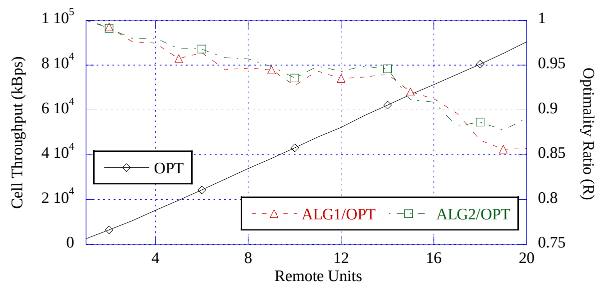

Fig. 8 – Cell throughput (left *y* axis) and $R_{\min}$ (right *y* axis) vs. number of RUs.

2) *Dominant-RU scenario*

We repeat the analysis in a more realistic scenario with one and three dominant RUs. Fig. 9 shows the carried load and *R* as a function of OL. We observe the same non-monotonic behavior as for the uniform case, with two noticeable differences: on one hand, the CT curve is not entirely flat after the knee, but instead keeps growing at a reduced slope for a wide range of offered loads. On the other hand, the width of the suboptimal region for *R* is reduced with respect to the uniform case. We analyze the behavior of $R_{\min}$ against the number of RUs. Fig. 10 confirms that the worst performance is achieved when the numbers of UEs and RUs are comparable.

Summing up, the accuracy of both Algorithm 1 and 2 appears to be excellent in both settings, and weakly influenced by the system parameters, at least within reasonable ranges of variation of the latter.

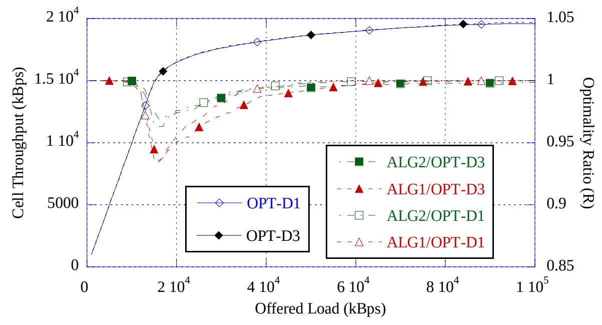

Fig. 9 – Cell throughput (left *y* axis) and optimality ratio (right *y* axis) vs. offered load, with one and three dominant RUs.

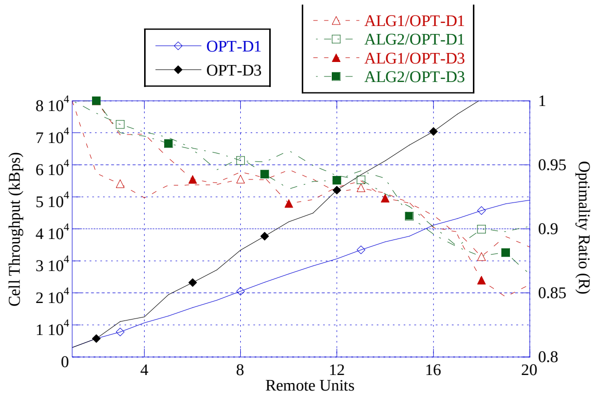

Fig. 10 – Cell throughput (left *y* axis) and $R_{\min}$ (right *y* axis) vs. number of RUs, dominant RUs.

3) *Impact of limited feedback*

We now evaluate the accuracy of Algorithms 1 and 2 when less information is available, due to limited feedback. Assume that, as it happens in practice, UEs only report *two* CQIs. Furthermore, the eNodeB assumes that $C_{i,j} = 0$ for the couples (*i,j*) for which it does not receive CQI reports. This reduces the possible matches between UEs and RUs. Fig. 11 and Fig. 12 report the graphs of the optimal cell throughput (left *y* axis) and the optimality ratio *R* (right *y* axis) in the case of uniform CQIs (to be compared with Fig. 4) and one dominant RU (to be compared with Fig. 9), respectively.

Quite counter-intuitively, the optimal cell throughput is marginally affected, and only in the dominant-RU scenario. This is because, since there are (on average) many UEs per RU, there is still a sufficient percentage of UEs with good channel conditions, hence each antenna may still achieve a high throughput by targeting these UEs. As far as the optimality of the two algorithms is concerned, the figures show the following: for Algorithm 2, $R_{\min}$ is practically unaltered in both scenarios. For Algorithm 1, there is instead a reduction in $R_{\min}$, slightly more evident in the dominant-RU scenario. However, in both cases, the $R_{\min}$ of Algorithm 1 is still well above 90%.

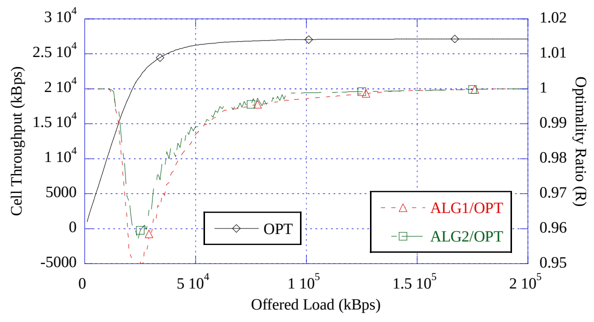

Fig. 11 – Cell throughput (left *y* axis) and optimality ratio (right *y* axis) vs. the offered load with uniform CQIs, when UEs report two CQIs (compare with Fig. 4).

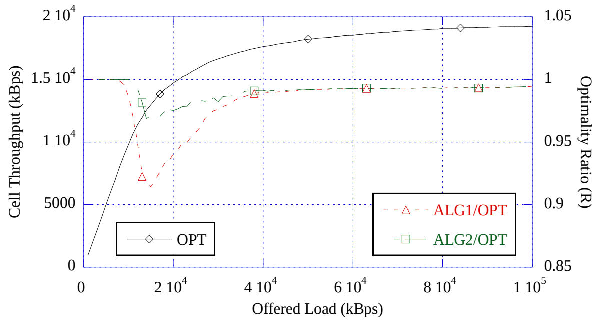

Fig. 12 – Cell throughput (left *y* axis) and optimality ratio (right *y* axis) vs. the offered load with one dominant RU, when UEs report two CQIs (compare with Fig. 9).

5.2 Time-based simulations

We now compare Algorithm 1, i.e., the simplest one, to the optimum, using traffic generators. Each scenario is simulated 5 times for 50s. We assume one dominant RU per UE and a k_i equal to 1, 2 or 4 with 40%, 40% and 20% probability. On each TTI, the dominant RU and the k_i may change with a 50% probability. Ideal reporting is assumed. Confidence intervals are not drawn when negligible. The traffic generators are the following:

1) *Downlink Web Traffic*: We simulate web traffic with a Pareto distributed on/off process (*on* period length: Shape=1.4, Scale=0.15s; *off* period length: Shape=1.2, Scale=0.5s). Packet size is 2KB, and inter-arrival times during *on* periods are 5ms. We simulate 4 RUs with 25 RBs per RU. Fig. 13 shows the average and 95th percentile delay for up to 100 UEs, a scenario which is close to the saturation point. Algorithm 1's performance is very close to the optimum.

2) *Video Traffic*: we use a Futurama trace [19] with mean rate=128KB/s and min/avg./max frame size equal to 95B/5KB/44KB, respectively. Fig. 14 shows the delay for up to 55 UEs. Algorithm 1 is nearly optimal until 48 UEs, and then exhibits significantly worse performance. Note, however, that beyond 50 UEs the cell is clearly saturated.

3) *VoIP Traffic*: This is an on/off traffic, whose durations are Weibull-distributed (*on* period: Shape=1.423s, Scale= 0.824s; *off* period: Shape=0.899s, Scale=1.089s). During *on* periods, 32B-packets are sent each 20ms. As the traffic is low-bandwidth, we simulate two RUs with three RBs per RU to keep the simulation time manageable. Fig. 15 shows the delay for up to 750 UEs. We observe that Algorithm 1 has a slightly larger 95th delay percentile, whereas the average delay almost matches the optimum.

4) *Mixed Traffic*: This configuration combines the three above traffics. We simulate 65% VoIP users, 22% video users, 13% Web users. Fig. 16 shows the 95th delay percentiles for up to 160 UEs, i.e. close to cell saturation. The figure shows that Algorithm 1 privileges web traffic to the detriment of video sources. Another interesting fact is that VoIP traffic is virtually unaffected by the simultaneous presence of competing traffic.

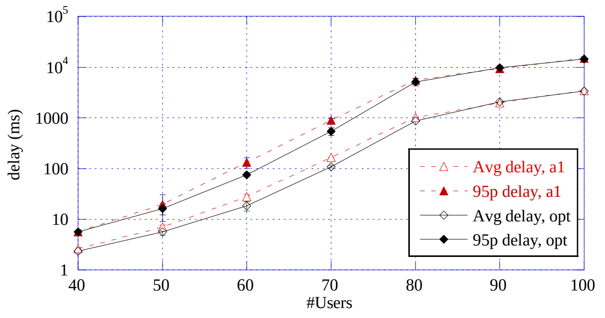

Fig. 13 – Delay vs. number of UEs, Web traffic

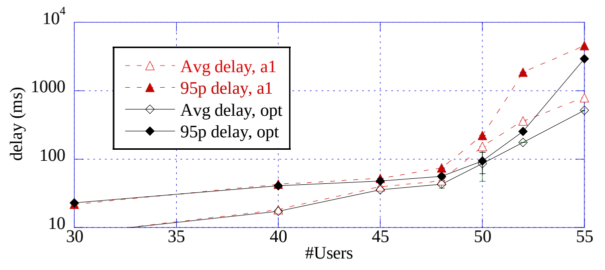

Fig. 14 – Delay vs. number of UEs, Video traffic

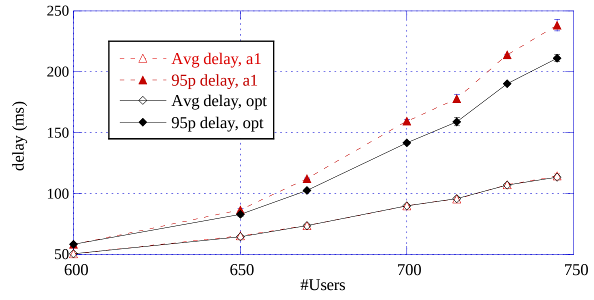

Fig. 15 – Delay vs. number of UEs, VoIP traffic

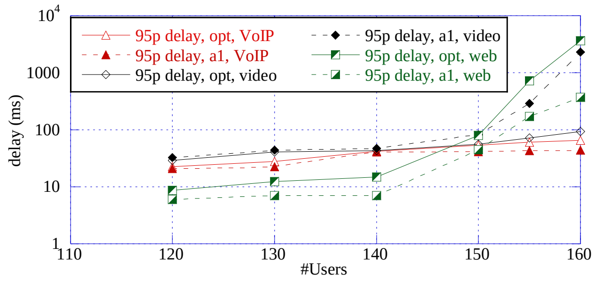

Fig. 16 – Delay vs. number of UEs, mixed traffic.

5.3 Throughput comparison for the two algorithms

The accuracy of Algorithms 1 and 2 has been shown to be comparable. There are cases, however, when the latter outperforms the former, e.g. when the number of UEs and RUs is similar and most UEs admit one stream only. Fig. 17 shows the cell throughput ratio of the two algorithms against the percentage of single-stream and 2-stream UEs. The scenario includes 20 full-buffer UEs and 10 RUs with 25 RBs each. The graph shows that a 7% gap exists when only single-stream UEs are in the cell.

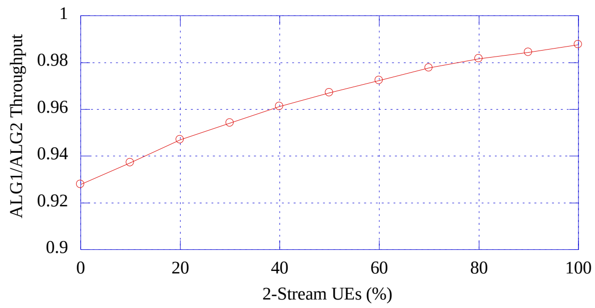

Fig. 17 – Throughput ratio between Algorithm 1 and Algorithm 2 as a function of the percentage of 2-stream UEs.

6 Related Work

The problem of MAC-level resource allocation on LTE-A with DAS is relatively new, which justifies the paucity of the literature on the subject. The works that are most directly related to ours are [4] and [23]. In the former, authors deal with the problem of allocating *transmission power* and *logical subbands* to downlink transmissions. A mixed integer-linear problem is formulated to compute a *fair* allocation, and heuristics are proposed to approximate the optimum. In the latter, instead, a linear optimization problem is proposed to match multiple RUs to UEs, hence assuming spatial multiplexing. However, the objectives and the models of [4] and [23] differ from ours. Neither take finite buffers into considerations, and [4] constrains a UE to receive transmissions from *one* antenna only. In [5] an algorithm to match antennas to UEs in the *uplink* direction is described. The algorithm is amenable to both Multiple-Input, Multiple Output (MIMO) and DAS systems. However, no resource allocation is proposed therein. The problem of antenna selection is dealt with in [22], where an algorithm for selecting the most appropriate antenna cluster is presented. Papers [6]-[8] deal with resource allocation under DAS in OFDM networks, i.e. where each antenna can serve a *single* UE, whereas [9] investigates capacity bounds in the uplink of DAS systems via link simulations. Authors of [10] evaluate both single-cell and multicell DAS systems employing two transmission techniques, *Maximum Ratio Transmission* (MRT) and *Selection Transmission* (ST), and make observations regarding where to deploy antennas in a cell. In [11] authors evaluate – via link level simulations – the performance increase in DAS systems employing known transmission schemes, namely MRT and *zero forcing beamforming* (ZFB). Finally, [12] studies multicast transmissions of layered video in a DAS deployment, taking into account power consumption and minimum rate requirements. Authors formulate the problem as a mixed integer-nonconvex optimization problem, and propose a iterative heuristic solution. No assumption on finiteness of buffers is mentioned in this work.

As a last remark, we observe that, if we constrain one RU to serve *only one* UE, as in [4] and [6]-[8], then the optimal solution can be computed in polynomial time, whether buffers are finite or not: it is in fact enough to set a cost for match (i, j) as:

$$c_{i,j} = C_{\max} \cdot B - \min\left\{C_{i,j} \cdot B_j , q_i\right\},$$

and run the Hungarian Algorithm accordingly. Obviously enough, since a whole frame normally accommodates much more traffic than a single UE's backlog, the above constraint leads to wasting resources.

7 Conclusions

In this paper we have described how to schedule RBs in an LTE-A cell using distributed antennas, so as to achieve the maximum throughput. We have shown that – under the assumption of finite buffers – the problem is both NP-hard and APX-hard, meaning that no polynomial-time heuristic can achieve bounded worst-case performance. We have presented two polynomial-time greedy heuristics, i.e. Algorithm 1 and Algorithm 2, which compute high-throughput allocations at different complexity. Algorithm 1 is linear in the number of UEs, whereas Algorithm 2 is cubic. Our evaluation has shown that the accuracy of the two heuristics is similar and quasi-optimal in practical cases, from both a throughput and a delay standpoint. This means that an accurate resource allocation is affordable in a real-life deployment.

Acknowledgements

The subject matter of this paper includes description of results of a research project carried out by the Dipartimento di Ingegneria dell'Informazione of the University of Pisa on behalf of, and funded by, Telecom Italia SpA, who reserves all proprietary rights in any process, procedure, algorithm, article of manufacture, or other result of said project herein described. The research described in this paper has been carried out when Giovanni Accongiagioco, Matteo Andreozzi and Daniele Migliorini were with the University of Pisa.

References

[1] 3GPP - TS 36.300: E-UTRA and E-UTRAN; Overall description; Stage 2

[2] 3GPP - TS 36.913 Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA)

[3] H. Holma, A. Toskala, "LTE for UMTS-OFDMA and SC-FDMA based Radio Access", Wiley, 2009

[4] P. Gong, K. Yu, Y. Wang, "Radio resource allocation for multiuser OFDMA distributed antenna systems", IEEE IC-NIDC, 2009.

[5] J. Choi, I. Sohn, S. Kim, K. B. Lee, "Efficient Uplink User Selection Algorithm in Distributed Antenna Systems", IEEE PIMRC 2007

[6] W. Xu, Z. He, K. Niu, "Opportunistic Packet Scheduling in OFDM Distributed Antenna Systems", WiCOM'09, 2009

[7] B. Yang, Y. Tang, "Heuristic Resource Allocation for Multiuser OFDM Distributed Antenna System with Fairness Constraints", ICCTA 2009

[8] L. Ling, T. Wang, Y. Wang, C. Shi, "Schemes of Power Allocation and Antenna Port Selection in OFDM Distributed Antenna Systems", IEEE VTC 2010-Fall

[9] P. Marsch, S. Khattak, G. Fettweis, "A Framework for Determining Realistic Capacity Bounds for Distributed Antenna Systems", Information Theory Workshop, 2006. ITW '06 Chengdu.

[10] J. Zhang, J.Andrews, "Distributed Antenna Systems with Randomness", IEEE Trans on Wireless Communications, 2008

[11] H. Zhu, S. Karachontzitis, D. Toumpakaris, "Low-complexity resource allocation and its application to distributed antenna systems", IEEE Wireless Communications, 2010

[12] L. Tian, Y. Zhou, Y. Zhang, G. Sun, J. Shi, "Resource allocation for multicas services in distributed antenna systems with quality of service guarantees", IET Comunications, 2012, vol.6, Issue 3, pp. 264-271

[13] M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman & Co., NY, 1979.

[14] H. W. Kuhn, "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, 2:83–97, 1955.

[15] R.E. Burkard, M. Dell'Amico, S. Martello, "Assignment Problems". SIAM, Philadelphia (PA.) 2009

[16] S. Martello, P. Toth, "Knapsack Problems: Algorithms and Computer Implementations". John Wiley & Sons, Ltd., 1990

[17] C. Chekuri, S. Khanna, "A PTAS for the multiple knapsack problem", in Proc. SODA'00, pp. 213–222, Philadelphia, PA, USA, 2000.

[18] IBM ILOG CPLEX Optimization Studio - http://tinyurl.com/33gy6co

[19] Video Trace Library - http://tinyurl.com/7e432wc

[20] M. Kottkamp, A. Roessler, J. Sclienz, LTE-Advanced Technology Introduction, Rohde & Schwarz white paper, 2012

[21] 3GPP – TS 36.211, June 2012, Technical Specification Group Radio Access Network; E-UTRA Physical Channels and Modulations, Release 10

[22] P Shang, G. Zhu, L. Tan, G. Su, T. Li, "Transmit Antenna Selection for the distributed MIMO Systems", 2009 Int. Conf. on Networks Security, Wireless Communications and Trusted Computing

[23] B. Yang, J. Xia, Y. Tang, H. Ba, F. Hua, "Optimal Downlink Resource Allocation in OFDMA Distributed Radio Access Network", Springer Advances in Technology and Management Vol. 165, 2012, pp. 393-401

8 Appendix

Proof of Property 5

We rewrite Algorithm 1 using the following data structures:

- A max heap, to store non-null allowances
- N circular lists UL_i, that link up to M non-null allowances related to UE i
- M circular lists AL_j, that link up to N non-null allowances related to antenna j

With reference to Fig. 18, in the `Init` function, computing allowances (lines 2-6) takes $O(N\cdot M)$ operations; building the circular lists takes $O(N\cdot M)$ operations as well, and building a max heap of $O(N\cdot M)$ allowances takes $O(N\cdot M\cdot\log(N\cdot M))$.

In the `Algorithm1` function, lines 15-20 have a constant cost. Furthermore, when these lines are executed, at least one of the conditions in the `if`s of lines 21-25 becomes true. The extractions required in line 21 (resp., 22 and 25) are performed at most *once* per RU (resp., UE) in a run. The entire cost of line 21 (resp., 22 and 25) in a run of the algorithm is thus $O(N\cdot M\cdot\log(N\cdot M))$.

The maximum number of iterations of the `while` cycle is upper bounded by both $O(N\cdot M)$ and $O(M\cdot B)$, since each element in the max heap is selected at most twice before being removed, and at least one RB is allocated on each iteration. The cost of a reheapification cycle in line 24 is $O(M\cdot\log(N\cdot M))$. Therefore, the overall complexity grows as:

$$\min\left\{\begin{matrix} O\left(N\cdot M^2\cdot\log(N\cdot M)\right), \\ O\left(M\cdot(M\cdot B+N)\cdot\log(N\cdot M)\right) \end{matrix}\right\},$$

which is the thesis.

□

```
1.  Function Init
2.    ∀i,j
3.      if B_j>0 and q_i>0 and k_i>0 then
4.        A_{i,j} = min{q_i, C_{i,j}}
5.      else
6.        A_{i,j} =0
7.    ∀i| A_{i,j}>0, link A_{i,j} in a circular list UL_i
8.    ∀j| A_{i,j}>0, link A_{i,j} in a circular list AL_j
9.    ∀i,j| A_{i,j}>0, sort A_{i,j} in max heap H
10. end function
11.
12. Function Algorithm1
13.   Init
14.   while not_empty(H) do
15.     (i,j) = top of max heap H
16.     if x_{i,j}=0 then k_i=k_i-1
17.     b=min {⌊q_i/A_{i,j}⌋, B_j}
18.     x_{i,j} = x_{i,j} + b
19.     q_i = q_i - b×A_{i,j}
20.     B_j = B_j - b
21.     if B_j = 0 then ∀i∈AL_j extract A_{i,j}
22.     if q_i = 0 then ∀j∈UL_j extract A_{i,j}
23.       else if q_i<A_{i,j} then
24.         ∀j∈UL_j, set A_{i,j}=min{A_{i,j},q_i} and re-heapify
25.     if k_i = 0 then ∀j∈UL_j | x_{i,j}=0, extract A_{i,j}
26. end function
```

Fig. 18 – Pseudo-code for Algorithm 1 using a max heap and circular lists.