Power efficient uplink scheduling in SC-FDMA: benchmarking by column generation

Yixin Zhao, Torbjörn Larsson, Di Yuan, Elina Rönnberg, Le and Lei

Journal Article

Yixin Zhao, Torbjörn Larsson, Di Yuan, Elina Rönnberg, Le and Lei, Power efficient uplink scheduling in SC-FDMA: benchmarking by column generation, Optimization and Engineering, 2016. 17(4), pp.695-725.
http://dx.doi.org/10.1007/s11081-015-9304-z

Power Efficient Uplink Scheduling in SC-FDMA: Benchmarking by Column Generation

Yixin Zhao[1,3], Torbjörn Larsson[1,4], Di Yuan[2], Elina Rönnberg[1], and Lei Lei[2]

[1]Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden
[2]Department of Science and Technology, Linköping University, SE-601 74 Norrköping, Sweden
[3]E-mail: yixin.zhao@liu.se
[4]E-mail: torbjorn.larsson@liu.se

Abstract

We study resource allocation in cellular systems and consider the problem of finding a power efficient scheduling in an uplink single carrier frequency division multiple access system (SC-FDMA). Due to the discrete nature of this problem and its computational difficulty, particularly in a real-time setting, the use of suboptimal algorithms is common practice. We aim at an effective way of gauging the performance of suboptimal algorithms by finding tight bounds on the global optimum.

Toward this end, we first provide a basic integer linear programming formulation. Then we propose a significantly stronger column-oriented formulation and a corresponding column generation method, as well as an enhanced column generation scheme. The latter extends the first scheme through the inclusion of a stabilization technique, an approximate column generation principle, and a tailored heuristic that is embedded in the column generation scheme to find high-quality though not necessarily global optimal solutions.

The computational evaluation demonstrates that compared with a poor performance by the integer linear programming formulation, the column generation method can produce near-optimal schedules that enable a sharp bounding interval. The enhanced column generation method significantly sharpens the bounding interval. Hence the column generation approach serves well for the purpose of benchmarking results for large-scale instances.

Keywords: localized SC-FDMA; stabilized column generation; power minimization; integer linear programming; uplink scheduling; matheuristic

1 Introduction

Growing demands on multimedia applications require the next generation cellular networks to transmit higher rate data to offer quality-of-service for the User Equipments (UEs). To provide high-speed data transmission for future wireless systems, Orthogonal Frequency Division Multiple Access (OFDMA) has been considered as a standard for its robustness to multipath fading, high spectrum flexibility and bandwidth scalability. While keeping these advantages of OFDMA, Single Carrier Frequency Division Multiple Access (SC-FDMA) provides much lower peak-to-average-power ratio, which mitigates the complexity of amplifier design and reduces power consumption of UE.

Unlike downlink, reducing UE energy consumption is the major concern in the uplink transmission for wireless communication systems, due to limited battery capacity of the UEs. Thus, SC-FDMA has been adopted as the standard uplink multiple access scheme in Long-Term Evolution (LTE), which is a 4G wireless communication standard developed by the Third Generation Partnership Project (European Telecommunications Standards Institute, 2009). We study the problem of power efficient scheduling of subcarriers in an uplink SC-FDMA system. Readers unfamiliar with wireless communications are referred to, for example, Tse and Viswanath (2005).

1.1 Related works

For 4G networks, most of the previous work that has dealt with optimizing power consumption and subcarrier allocation is for OFDMA systems. Algorithms regarding resource allocation with the objective of maximizing the utility or minimizing the power consumption have been addressed in Cheong et al. (1999), Miao et al. (2010), Lei et al. (2012), and Gao and Wang (2011).

In SC-FDMA, there are two schemes for allocating subcarriers to multiple UEs: interleaved and localized (Myung et al., 2006). With the former, each UE transmits data by subcarriers spread over the entire bandwidth. In the latter, each UE is restricted to occupy a set of consecutive subcarriers in the frequency domain. Primarily due to this adjacency restriction, the previously developed scheduling algorithms for OFDMA cannot be directly applied in localized SC-FDMA.

From the point of view of optimization, the adjacency constraint is the main difference between SC-FDMA and OFDMA, and makes the resource allocation optimization more challenging. The NP-hardness of incorporating the adjacency

constraint into scheduling algorithms with respect to different performance measures has been proven in Wong et al. (2009) and Ahmad and Assaad (2011), and most proposed algorithms are greedy-based heuristics ones (Kim et al., 2010; Lei et al., 2013a,b; Wong et al., 2009).

Different system performance measures for SC-FDMA scheduling and subcarrier allocation are presented in Lee et al. (2009), Wong et al. (2009), Ahmad and Assaad (2011), and Safa and Tohme (2012). Maximizing the sum utility, known as the proportional fair criterion, in the time and frequency domain, is presented in Lee et al. (2009), where some sub-algorithms for assigned UEs are also proposed. In Wong et al. (2009), the authors formulated the sum utility maximization problem with subcarrier-adjacency restriction in SC-FDMA systems as a Binary-Integer Programming (BIP) problem. In Ahmad and Assaad (2011), the authors transformed the BIP problem in Wong et al. (2009) into a canonical dual problem, and based on this approach, they developed a polynomial-time complexity, but suboptimal, algorithm to solve the power minimization problem in SC-FDMA uplink. The problem of minimizing the number of allocated subcarriers meeting the given demand, is considered in Safa and Tohme (2012), as a measure for the performance of the existing greedy-based heuristic algorithms. Comparatively, the problem of minimizing power efficiency has been much less studied.

We study subcarrier-consecutive power efficient scheduling in SC-FDMA systems over multiple transmission time slots, assuming full knowledge of the communication channel for the entire scheduling period and with the objective to minimize the total power consumption. As mentioned above, several greedy-based heuristics have been proposed in the literature. These heuristics however lack quality guarantees and the schedules produced can be far from being optimal. We approach the scheduling problem with static formulation with the purpose of computationally delivering global optimality or a tight bounding interval confining global optimum. This is not a trivial task in view of problem complexity.

On the methodology side, we adopt the notion of column-oriented formulations and column generation (e.g., Vanderbeck et al., 2005), to achieve our objective of gauging the performance of energy efficiency of SC-FDMA scheduling from an optimization standpoint. The column generation methodology overcomes the scalability issue and is able to produce a global optimum or tight bounds on a global optimum. The principle of column generation depends heavily of linear programming theory, and the reader who is not familiar with linear programming is referred to, for example, Bertsimas and Tsitsiklis (1997).

1.2 Column generation

Column generation is a method for linear programs that have too many variables (i.e., columns) to be considered explicitly, but the variables have a common structure that allows them to be created systematically. Based on the observation that for such problems the vast majority of the variables have no significance at optimum, that is, they attain value zero, only a subset of variables are of actual interest when solving the problem. Column generation leverages this observation to create only the variables which have the potential to improve the objective value, by using dual multipliers.

Column generation alternates between solving a restricted version of the linear program, counting only a small subset of columns and referred to as the restricted master problem, and a subproblem, also called column generation or pricing problem (Dantzig and Wolfe, 1960). The subproblem identifies columns that are profitable to add in the restricted master problem, and after solving this problem, the value of the dual variables are sent to the pricing problem. For a minimization problem, the restricted master problem provides upper bounds on the optimal value, while the dual solution and the pricing problem provide lower bounds.

Many discrete optimization problems admit different integer programming formulations, some being less straightforward than others. Among the less straightforward ones are the column-oriented formulations, which include huge numbers of variables and are suited for column generation. Column-oriented formulations and column generation have proven to be one of the most successful approaches for solving large-scale discrete optimization problems. The linear programming relaxation of such a formulation has the potential to yield a tighter bound on the global optimum than the linear programming relaxation of a straightforward integer linear programming formulation (e.g. Lübbecke and Desrosiers, 2005, Section 5.3.1), provided that the pricing problem is a discrete optimization problem which is NP-hard while it can still be solved with sufficient practical efficiency. A hybrid between column generation and branch-and-bound, called branch-and-price, has become a very powerful technique for solving a wide range of industrial discrete optimization problems to optimality or to near optimality, most efficiently for routing and scheduling problems (e.g., Barnhart et al., 1998; Desrosiers et al., 1995).

It is well known that basic column generation approaches often suffer from computational difficulties such as degeneracy and tailing-off, which often increase their computational cost dramatically. Moreover, instability of the behaviour in the dual space is frequent and harmful. This instability manifests itself as oscil-

lations in the values of the dual variables, resulting in slow convergence. Further, while the upper bound given by the restricted master problem in non-increasing, the instability causes the lower bound computed for each dual point to oscillate heavily, so that the best lower bound is improved only occasionally. The instability also worsens the quality of the columns that are generated. It can however be dealt with by using stabilization techniques (e.g., Marsten et al., 1975; du Merle et al., 1999; Larsson et al., 2004).

1.3 Contributions and paper outline

We first formulate the power minimization uplink SC-FDMA scheduling problem as an integer linear program. However, this basic formulation does not admit optimality within reasonable computing effort. We therefore propose a column-oriented formulation of the problem. Lower bounds on the optimal value are found through column generation, while upper bounds are found through the solution of integer versions of the restricted master problem. We further present an enhanced column generation scheme, which includes a stabilization technique, an approximate column generation, that is used initially, and an accelerated upper bounding based on a local search procedure that converts generated columns into feasible solutions.

The pricing problem in our application of column generation is a discrete optimization problem, and the possibility to reach a tight lower bound on the global optimum is the primary motive for studying this approach. Our approach can be characterized as a matheuristic and we obtain good numerical results without having to resort to using branch-and-bound. Our key contribution is a global optimization approach that makes it computationally affordable to gauge the performance of any proposed suboptimal scheduling algorithm with unknown solution quality. The research presented in this paper is a continuation of the preliminary work first published in Zhao et al. (2013).

This paper is organized as follows. Section 2 describes the system model and gives an integer programming formulation. Section 3 gives the column-oriented formulation and a standard column generation solution scheme. Section 4 provides the enhanced column generation method. In Section 5, we illustrate the numerical results of performance evaluation and finally Section 6 summarizes the key conclusions and give suggestions for future research.

2 Problem definition

We here provide the system specifications and give an integer programming formulation of the optimization problem to be solved.

2.1 System model

We consider an uplink SC-FDMA multi-UE system in a single cell with N narrowband subcarriers and U UEs. The base station, as a scheduler, assigns consecutive subcarriers to each UE at each time slot t to minimize the total power consumption of the UEs over T time slots.

We assume full knowledge of the communication channel for the entire scheduling period. This is in line with the solution approach we propose, namely, the focus is to obtain a powerful tool to deliver either the global optimum or a tight bounding interval. With such a tool, one is able to gauge the performance of any other suboptimal scheduling algorithms that may have to deal with unknown and stochastic channel conditions. That is, after applying a suboptimal algorithm, one can compute the bounds with known data for benchmarking, in order to tell if the algorithm performs well, or there is still significant room for improvement. Moreover, with the anticipatory networking paradigm that is expected to be part of next-generation communication systems, network is able to predict the changing conditions of users, and in such cases our solution approach is applicable in a rolling horizon fashion.

For localized allocation in SC-FDMA, the subcarriers assigned to each UE must be consecutive in the spectrum. Due to this restriction, candidate subcarriers can be grouped into candidate subcarrier blocks. Note that candidate blocks are obtained by considering all subsets of consecutive subcarriers, and therefore each subcarrier is part of multiple candidate blocks. We use B to denote the number of nonempty and consecutive subcarrier blocks. Clearly, the block sizes range between zero and N, and the value of B can be derived as follows. A subcarrier block forms a consecutive interval $[n, n']$, where $1 \leq n \leq n' \leq N$. For given n, there are $N - (n-1)$ possible values of n'. Letting n take values $1, \ldots, N$, we obtain $N + (N-1) + \cdots + 1$ choices in total, giving $B = N + (N-1) + \cdots + 1 = N(N+1)/2$. Let N_b be the set of subcarriers included in block $b = 0, 1, \ldots, B$, where $b = 0$ is indexing the empty block, containing no subcarriers at all. We treat the subcarrier blocks as the basic resource allocation units in the frequency domain.

In addition, there are two constraints for power allocation in SC-FDMA uplink. The transmission power for each UE u should be less than a power level limit, denoted by P_u^{max}, with equal power allocation to the subcarriers for the UE (European Telecommunications Standards Institute, 2009). For each individual subcarrier n, the power can not exceed a given peak power level P_n^{max}. Let r_{ub}^t denote the instantaneous data rate achieved by allocating subcarrier block b to UE u at time slot t with power $\min\{P_u^{max}/|N_b|, P_n^{max}\}$ on each subcarrier, that is, with total power $p_{ub} = \sum_{n \in N_b} \min\{P_u^{max}/|N_b|, P_n^{max}\}$ for the block. If block b is empty, the power is zero and the block rate is zero accordingly. For nonempty blocks, Shannon's capacity function (e.g. Tse and Viswanath, 2005) gives the block date rate

$$r_{ub}^t = \sum_{n \in N_b} \log_2 \left(1 + \min\left\{P_u^{max}/|N_b|, P_n^{max}\right\} \gamma_{unt}\right),$$

where γ_{unt} is the channel gain-to-noise ratio for UE u on subcarrier n at time slot t.

Each UE has a rate demand d_u over the T time slots. This demand is satisfied by allocating subcarrier blocks to the UE, leading to a given power consumption. The overall problem is then to make the allocation of subcarrier blocks to the UEs such that all demands will be fulfilled while total power consumption is minimized.

2.2 Integer programming formulation

We begin by providing a list of used notations.

Parameters:

- U = number of UEs.
- N = number of subcarriers.
- T = number of time slots.
- B = number of possible combinations of consecutive subcarriers within the same time slot.
- d_u = the rate demand for UE u, where $u = 1, \ldots, U$.

- r_{ub}^t = the instantaneous block rate for UE u on subcarrier block b at time slot t, where $u = 1, \ldots, U,\ b = 0, \ldots, B,\ t = 1, \ldots, T$.
- p_{ub} = the power consumption for UE u with subcarrier block allocation b, where $u = 1, \ldots, U,\ b = 0, \ldots, B$.
- $a_{nb} = \begin{cases} 1 & \text{if subcarrier } n \text{ is contained in subcarrier block } b \\ 0 & \text{otherwise,} \end{cases}$
  where $n = 1, \ldots, N$ and $b = 0, \ldots, B$.

Variables:

- $x_{ub}^t = \begin{cases} 1 & \text{if subcarrier block } b \text{ is allocated to UE } u \text{ at time slot } t \\ 0 & \text{otherwise,} \end{cases}$
  where $u = 1, \ldots, U,\ b = 0, \ldots, B,\ t = 1, \ldots, T$.

The power efficient scheduling problem for the SC-FDMA uplink system can then be stated as the following Integer Program (IP).

$$\text{minimize} \quad \sum_{u=1}^{U} \sum_{b=0}^{B} \sum_{t=1}^{T} p_{ub} x_{ub}^t \tag{1}$$

$$\text{subject to} \quad \sum_{b=0}^{B} \sum_{t=1}^{T} r_{ub}^t x_{ub}^t \geq d_u \quad u = 1, \ldots, U \tag{2}$$

$$\sum_{u=1}^{U} \sum_{b=0}^{B} a_{nb} x_{ub}^t \leq 1 \qquad \begin{matrix} n = 1, \ldots, N, \\ t = 1, \ldots, T \end{matrix} \tag{3}$$

$$\sum_{b=0}^{B} x_{ub}^t = 1 \qquad \begin{matrix} u = 1, \ldots, U, \\ t = 1, \ldots, T \end{matrix} \tag{4}$$

$$x_{ub}^t \in \{0, 1\} \qquad \begin{matrix} u = 1, \ldots, U, \\ b = 0, \ldots, B, \\ t = 1, \ldots, T \end{matrix} \tag{5}$$

Objective function (1) is to minimize the total consumed power of all the UEs over all time slots. Constraint (2) requires that the allocation of subcarrier blocks to each UE over all time slots satisfies the UE's rate demand, while constraint (3) states that each subcarrier can be assigned to at most one UE in each time slot. If any block containing a certain subcarrier is allocated to a UE in a time slot, then all the subcarriers within this block become occupied, and no other UE can in this

time slot be given any block that contains any of these subcarriers. This effect is achieved by the construction of (3). Constraint (4) ensures that exactly one subcarrier block is allocated to each UE at each time slot, in order to guarantee the consecutiveness of the assigned subcarriers. It prevents that nonadjacent blocks are allocated to the same UE at the same time slot, while for adjacent blocks there is a corresponding single block alternative by the construction of the blocks. If $x^t_{u0} = 1$ holds, then the empty block is assigned.

This model for power efficient scheduling of the SC-FDMA uplink system has the disadvantage of a Linear Programming (LP) relaxation that is not particularly strong in providing tight lower bounds to the global optimum. Finding a schedule that is optimal, or even only near-optimal, using this model is therefore computationally demanding (to be seen in the numerical results in Section 5).

3 Column-oriented formulation

As an alternative to the formulation IP in the previous section, we propose to use column generation (e.g., Lübbecke and Desrosiers, 2005; Wilhelm, 1995), which decomposes the original problem into two optimization problems that are solved alternately. Specifically, one alternates between the solution of a restricted master problem containing only a subset of the variables, and the use of its dual solution in a pricing problem to find new and profitable variables.

The approach presented in this section represents standard practice of the theory of column generation, based on a column-oriented reformulation of the IP. Results and notions for improving the column generation efficiency, tailored to the specific problem at hand, will be detailed in Section 4.

3.1 The master problem

To apply column generation to the problem of power efficient scheduling in a SC-FDMA uplink system, we first state the column-oriented formulation. It is based on decision variables that, for each UE, correspond to sequences of subcarrier blocks over time slots, with exactly one subcarrier block chosen per time slot, under the additional requirement that these subcarrier blocks together shall satisfy the rate demand for the UE. Hence, each decision variable corresponds to one feasible solution to the restrictions (2), (4) and (5). We start by introducing more notations.

Parameters:

- $S_u =$ number of feasible subcarrier block sequences for UE u over all time slots, where $u = 1, \ldots, U$.
- $e^t_{ubs} = \begin{cases} 1 & \text{if subcarrier block } b \text{ is contained in the feasible subcarrier block} \\ & \text{sequence } s \text{ for UE } u \text{ at time slot } t, \\ 0 & \text{otherwise,} \end{cases}$
  where $u = 1, \ldots, U,\ b = 0, \ldots, B,\ s = 1, \ldots, S_u$, and $t = 1, \ldots, T$.

Variables:

- $x_{us} = \begin{cases} 1 & \text{if feasible subcarrier block sequence } s \text{ is assigned to UE } u, \\ 0 & \text{otherwise,} \end{cases}$
  where $u = 1, \ldots, U$, and $s = 1, \ldots, S_u$.

Clearly, for each user exactly one of the variables x_{us} shall take the value one. The decision variables of the problem IP depend of the variables x_{us} according to

$$x^t_{ub} = \sum_{s=1}^{S_u} x_{us} e^t_{ubs}, \quad u = 1, \ldots, U,\ \ b = 1, \ldots, B,\ \ t = 1, \ldots, T.$$

Inserting this relationship into IP, we obtain the Integer Programming Master (IPM) problem

$$\text{minimize} \qquad \sum_{u=1}^{U} \sum_{s=1}^{S_u} \left(\sum_{t=1}^{T} \sum_{b=0}^{B} e^t_{ubs} p_{ub} \right) x_{us} \tag{6}$$

$$\text{subject to} \qquad \sum_{s=1}^{S_u} x_{us} = 1 \qquad u = 1, \ldots, U \tag{7}$$

$$\sum_{u=1}^{U} \sum_{s=1}^{S_u} \left(\sum_{b=0}^{B} e^t_{ubs} a_{nb} \right) x_{us} \leq 1 \qquad \begin{matrix} n = 1, \ldots, N \\ t = 1, \ldots, T \end{matrix} \tag{8}$$

$$x_{us} \in \{0, 1\} \qquad s = 1, \ldots, S_u, u = 1, \ldots, U. \tag{9}$$

Objective function (6) minimizes the consumed power for all the UEs with respect to the choice of sequences of subcarrier blocks over all time slots for all the UEs. Constraint (7) ensures that exactly one subcarrier block sequence is selected for each UE. Constraint (8) states that at each time slot, any subcarrier should be

assigned to at most one UE. Note that since the parameters a_{nb} and e^t_{ubs} indicate whether subcarrier n is contained in subcarrier block b, and whether subcarrier block b is contained in subcarrier block sequence s, respectively, we then accordingly have that $\sum_{b=0}^{B} e^t_{ubs} a_{nb}$ identifies whether subcarrier block sequence s for UE u includes subcarrier n at time slot t or not. Note that constraints (2) and (4) are embedded in the definition of the variables of the column-oriented formulation together with constraint (7).

Since the number of feasible block sequences is huge, although finite, the column-oriented formulation contains a huge number of variables. This means that it is in practice virtually impossible to set up the model. In a column generation approach for the model, one considers the linear programming relaxation and solves a restricted master problem which contains only a subset of the variables.

Suppose that s_u feasible block sequences are available for UE $u = 1, \ldots, U$, where $s_u \leq S_u$. Typically $s_u \ll S_u$ holds. We refer to the corresponding restricted version of IPM as RIPM. Let LPM denote the Linear Programming Master problem, that is, the continuous relaxation of IPM. Then the resulting Restricted Linear Programming Master (RLPM) problem is as follows.

$$\text{minimize} \quad \sum_{u=1}^{U} \sum_{s=1}^{s_u} \left(\sum_{t=1}^{T} \sum_{b=0}^{B} e^t_{ubs} p_{ub} \right) x_{us} \tag{10}$$

$$\text{subject to} \quad \sum_{s=1}^{s_u} x_{us} = 1 \qquad u = 1, \ldots, U \tag{11}$$

$$\sum_{u=1}^{U} \sum_{s=1}^{s_u} \left(\sum_{b=0}^{B} e^t_{ubs} a_{nb} \right) x_{us} \leq 1 \qquad \begin{matrix} n = 1, \ldots, N \\ t = 1, \ldots, T \end{matrix} \tag{12}$$

$$x_{us} \geq 0 \qquad s = 1, \ldots, s_u, u = 1, \ldots, U \tag{13}$$

Even though the above formulation differs just slightly compared to IPM, we give it for the sake of rigor and for the convenience of introducing the dual variables. The problem RLPM gives a feasible solution to LPM, and thus an upper bound to the optimal value of LPM. Let v_u represent the dual variable of constraint (11) and let y_{nt} represent the dual variable of constraint (12). These dual variables are instrumental for the generation of additional feasible block sequences through the solution of a pricing problem.

3.2 The pricing problem

By using the optimal dual values for the restricted master problem, denoted $\bar{v}_u$ and $\bar{y}_{nt}$, the reduced cost for an arbitrary column in LPM is

$$\begin{aligned}\bar{c}_{us} &= \sum_{t=1}^{T}\sum_{b=0}^{B} e_{ubs}^{t} p_{ub} - \sum_{n=1}^{N}\sum_{t=1}^{T} \bar{y}_{nt} \left(\sum_{b=0}^{B} e_{ubs}^{t} a_{nb} \right) - \bar{v}_u \\ &= \sum_{t=1}^{T}\sum_{b=0}^{B} \left(p_{ub} - \sum_{n=1}^{N} \bar{y}_{nt} a_{nb} \right) e_{ubs}^{t} - \bar{v}_u.\end{aligned} \tag{14}$$

Finding the most profitable column for each UE u amounts to obtaining the minimum value of $\bar{c}_{us}$ over $s = 1, \ldots, S_u$, which is equivalent to solving the linear binary problem

$$\text{minimize} \qquad \sum_{t=1}^{T}\sum_{b=0}^{B} \left(p_{ub} - \sum_{n=1}^{N} \bar{y}_{nt} a_{nb} \right) e_{ub}^{t} - \bar{v}_u \tag{15}$$

$$\text{subject to} \qquad \sum_{t=1}^{T}\sum_{b=0}^{B} r_{ub}^{t} e_{ub}^{t} \geq d_u \tag{16}$$

$$\sum_{b=0}^{B} e_{ub}^{t} = 1 \qquad t = 1, \ldots, T \tag{17}$$

$$e_{ub}^{t} \in \{0, 1\} \qquad t = 1, \ldots, T,\ b = 0, \ldots, B, \tag{18}$$

since each column in LPM corresponds to one feasible solution to the restrictions (2), (4) and (5). In this so-called pricing or column generation problem, the binary variable e_{ub}^{t} is indicating whether subcarrier block b in time slot t is included in a subcarrier block sequence for UE u or not. The optimal values of these variables define a feasible subcarrier block sequence for UE u, and correspond to a generated column to be included in RLPM, provided that its reduced cost is negative.

Note that the subcarrier block sequences for all UEs obtained from this pricing problem, when sent to the master problem, can violate constraint (8), that is, the same subcarrier can be allocated to two different users in the same time slot.

On one hand, the pricing problem is a binary program, which is difficult to solve in general, nevertheless it can be solved by, for example, branch-and-bound. On the other hand, it is recognized as a multiple-choice knapsack problem with

$(B+1)T$ variables and T multiple-choice constraints, and it can be solved comparatively efficiently by exploiting the special structure (Pisinger, 1995).

The feasible set of the linear programming relaxation of the pricing problem does in general not have the integrality property, and therefore the LPM is as least as strong as LP with respect to the ability of bounding the optimal value of the problem IP; see, e.g., Lübbecke and Desrosiers (2005), Section 5.3.1. Letting $v(\cdot)$ denote the optimal value of an optimization problem, the bounding properties are summarized in the proposition below.

Proposition 1 $v(LP) \leq v(LPM) \leq v(IP)$

Because of the lack of integrality property, $v(\text{LPM}) > v(\text{LP})$ is likely to hold.

The column generation procedure provides lower bounds on the optimal value of LPM, and therefore also on the optimal value of IP, in the usual way (e.g., Wolsey, 1998, pp 188–189). For future reference, we here establish the lower bound. By making a Lagrangian relaxation and using strong linear programming duality for RLPM we obtain that

$$
\begin{aligned}
&v(\text{LPM}) \\
&\geq \min \sum_{u=1}^{U}\sum_{s=1}^{S_u}\left(\sum_{t=1}^{T}\sum_{b=0}^{B} e_{ubs}^{t} p_{ub}\right) x_{us} - \sum_{n=1}^{N}\sum_{t=1}^{T} \bar{y}_{nt}\left(\sum_{u=1}^{U}\sum_{s=1}^{S_u}\left(\sum_{b=0}^{B} e_{ubs}^{t} a_{nb}\right) x_{us} - 1\right) \\
&\quad s.t.\ \ (11),\ \ (13) \\
&= \sum_{n=1}^{N}\sum_{t=1}^{T} \bar{y}_{nt} + \min \sum_{u=1}^{U}\sum_{s=1}^{S_u} \left(\overline{c}_{us} + \overline{v}_u\right) x_{us} \\
&\quad\quad s.t.\ \ (11),\ \ (13) \\
&= \sum_{n=1}^{N}\sum_{t=1}^{T} \bar{y}_{nt} + \sum_{u=1}^{U} \overline{v}_u + \sum_{u=1}^{U} \min_{s=1,\ldots,S_u} \overline{c}_{us} \\
&= v(\text{RLPM}) + \sum_{u=1}^{U} \min_{s=1,\ldots,S_u} \overline{c}_{us}.
\end{aligned}
$$

3.3 Summary of column generation

A column generation approach for the SC-FDMA problem is obtained by combining column generation that provides lower bounds to $v(\text{IP})$ and new columns

to the restricted master problem, with the solution of the integer version of the restricted master problem, which provides upper bounds to v(IP).

We use UBD and LBD to denote the upper and lower bounds. The UBD is calculated as v(RIPM). The lower bound established in the previous section is not monotonically increasing, so the highest value of v(RLPM)$+\sum_{u=1}^{U} \min_{s=1,\ldots,S_u} \overline{c}_{us}$ found during the iterations performed so far yields the lower bound LBD on v(LPM). As v(LPM) $\leq$ v(IP) holds, this is also an LBD on v(IP). (The initial lower bounding is a special case. For the zero dual solution, the pricing problems constitute a relaxation of the problem IP and therefore $\sum_{u=1}^{U} \min_{s=1,\ldots,S_u} \overline{c}_{us}$ then gives a lower bound.) The column generation terminates when $\overline{c}_{us} \geq 0$ holds for all columns generated in an iteration. The complete algorithm is summarized in Algorithm 1 below. We solve RIPM frequently, mainly in order to compare with the performance of the enhanced column generation scheme, to be presented in the next section. Besides, the computational cost of solving each RIPM is actually very low, as shown later.

As shown in the above section, the lower bound provided by the column generation method is going to be at least as strong as, and likely also stronger than, that of the linear programming relaxation of the IP formulation. The numerical results presented in Section 5 show that the former lower bound will indeed be stronger than the latter, and that the difference is significant.

```
initialization;
set dual solution to zero;
solve all pricing problems and update LBD ;
while (UBD − LBD)/LBD > ε do
    solve the RLPM to obtain a dual solution;
    solve the RIPM to get UBD;
    solve all pricing problems and update LBD ;
    if profitable columns are found then
        add profitable columns to the RLPM;
    else
        break;
    end
end
```

Algorithm 1: Column Generation

4 Enhanced column generation

Although the lower bounds produced by column generation are significantly better than those of the IP, there is still some space for improvements. We employ two strategies for speeding up the process of column generation and lower bounding, where the first is to stabilize the master problem by means of dual step size restrictions, and the second is approximate column generation, described in Sections 4.1 and 4.2, respectively. In order to improve the upper bounding quality, a local search is applied to the schedules found by the pricing problems, with the goal of converting them to a set of power efficient and non-conflicting schedules; this is described in Section 4.3.

4.1 Master problem stabilization

As is well known, the dual equivalent of column generation is a cutting plane procedure. It is known that in cutting plane procedures the consecutive iterates can be distant from each other, which makes such procedures instable. This is also the case for the dual cutting plane interpretation of column generation (e.g., Hiriart and Lemaréchal, 1993, Ch.15). Hence, the dual solutions obtained in consecutive iterations tend to vary significantly, which also leads to slow convergence in the primal problem. To improve the stability, the column generation scheme presented in Section 3 is modified to include a boxstep restriction (Marsten et al., 1975) on the individual dual variables, so that each new dual point is kept close to the previous one (see also, e.g., du Merle et al., 1999; Larsson et al., 2004; Westerlund et al., 2006).

For the dual problem of the RLPM presented in Section 3.1, we choose to include box constraints only for the dual variables y_{nt}, since the values of the dual variables v_u are uniquely determined by the values of y_{nt} due to the structure of the constraints. For each dual variable y_{nt}, we introduce a dual box with a non-positive center, denoted by $\hat{y}_{nt}$, and with the lower and upper bounds of the box defined by $y_{nt}^{lb} := \hat{y}_{nt} - \delta$ and $y_{nt}^{ub} := \hat{y}_{nt} + \delta$, respectively. Denote the optimal values of the dual variables by $\overline{y}_{nt}$ and $\overline{v}_u$. We then update the box according to

$$\hat{y}_{nt} = \begin{cases} \hat{y}_{nt} & \text{if} \quad y_{nt}^{lb} < \overline{y}_{nt} < y_{nt}^{ub} \\ y_{nt}^{lb} & \text{if} \quad \overline{y}_{nt} = y_{nt}^{lb} \\ y_{nt}^{ub} & \text{if} \quad \overline{y}_{nt} = y_{nt}^{ub}. \end{cases}$$

The boxstep restricted dual problem, or the dually equivalent Stabilized Restricted Linear Programming Master (SRLPM) problem, will generally not yield feasible solutions and upper bounds to the optimal value of LPM, since it is both a relaxation of the dual of LPM and boxstep restricted. If, however, the boxstep restrictions are not binding at an optimal solution, so that they are in fact redundant, then the optimal value is indeed an upper bound to LPM. Further, the stabilized column generation terminates first when $\overline{c}_{us} \geq 0$ holds for all columns generated in an iteration and the boxstep restrictions are not active.

Although the problem SRLPM does in general not produce upper bounds to LPM, the lower bounding calculation used in column generation can be extended to stabilized column generation, if *v*(RLPM) is replaced by *v*(SRLPM), as stated below.

Proposition 2

$$v(\textit{LPM}) \geq v(\textit{SRLPM}) + \sum_{u=1}^{U} \min_{s=1,\ldots,S_u} \overline{c}_{us}$$

This result follows from arguments analogue to those used at the end of Section 3.2. By tracing the arguments used there, it is evident that lower bounds can be calculated in the same way, no matter how the dual solution is generated.

4.2 Approximate pricing

The pricing problem presented in Section 3.2 is rather computationally demanding. A possible approach is to apply the tailored solution method of Pisinger (1995). We here instead give a method for quickly finding approximate solutions to the pricing problem. This approximate pricing will eventually fail to find columns with negative reduced costs, and it is therefore used only in some initial column generation iterations. We here consider the linear programming relaxation of the pricing problem, given by replacing the restriction $e^t_{ub} \in \{0, 1\}$ with $e^t_{ub} \in [0, 1]$.

Proposition 3 *An optimal solution to the linear programming relaxation of the pricing problem* (15)–(18) *is either integral, in which case it solves the pricing problem, or it is a non-trivial convex combination of* exactly two *integer points, which both satisfy constraints* (17) *and* (18)*, while at least one is also feasible in constraint* (16)*.*

Proof: If an optimal solution to the linear programming relaxation of the pricing problem is integral, then it obviously solves the pricing problem.

Consider the case when a linear programming solution is non-integral. This solution clearly contains $T+1$ basic variables. Suppose that the slack variable in constraint (16) is basic, so that T out of the variables e^t_{ub} are basic. Since each of the constraints (17) must contain at least one non-zero, and therefore basic, variable, it can be concluded that exactly one of the variables e^t_{ub} is non-zero in each of the constraints (17), which implies that the solution is integral. This contradicts that the solution is non-integral and we conclude that the slack variable in constraint (16) is non-basic.

Hence, $T+1$ out of the variables e^t_{ub} are basic. Again using that each of the constraints (17) must contain at least one non-zero variable, it follows that exactly one of these constraint, say for $t=\bar{t}$, contains exactly two basic variables, while each of the others contains exactly one. For $t \neq \bar{t}$, the solution is integral. For $t=\bar{t}$, exactly two of the variables take fractional values, because in case of degeneracy the solution would become integral. Denote the linear programming solution by $\overline{e}^t_{ub}$, for all t, u and b. Assume that $\overline{e}^t_{ub_1}$ and $\overline{e}^t_{ub_2}$ are the positive values of the two fractional variables, with $\overline{e}^{\bar{t}}_{ub_1} + \overline{e}^{\bar{t}}_{ub_2} = 1$. Define the two integer solutions

$$\hat{e}^t_{ub} = \begin{cases} \overline{e}^t_{ub} & \forall b, & t \neq \bar{t} \\ 1 & b = b_1, & t = \bar{t} \\ 0 & b \neq b_1, & t = \bar{t} \end{cases} \quad \text{and} \quad \tilde{e}^t_{ub} = \begin{cases} \overline{e}^t_{ub} & \forall b, & t \neq \bar{t} \\ 1 & b = b_2, & t = \bar{t} \\ 0 & b \neq b_2, & t = \bar{t}, \end{cases}$$

which are both feasible with respect to the constraints (17) and (18). Then the linear programming solution can be expressed as the convex combination

$$\overline{e}^t_{ub} = \overline{e}^{\bar{t}}_{ub_1} \hat{e}^t_{ub} + \overline{e}^{\bar{t}}_{ub_2} \tilde{e}^t_{ub} \qquad b = 0, \ldots, B,\ t = 1, \ldots, T.$$

Inserting this expression in (16) and using that the slack variable is non-basic, gives

$$\overline{e}^{\bar{t}}_{ub_1} \sum_{b,t} r^t_{ub} \hat{e}^t_{ub} + \overline{e}^{\bar{t}}_{ub_2} \sum_{b,t} r^t_{ub} \tilde{e}^t_{ub} = d_u.$$

Since $\overline{e}^{\bar{t}}_{ub_1}, \overline{e}^{\bar{t}}_{ub_2} > 0$ and $\overline{e}^{\bar{t}}_{ub_1} + \overline{e}^{\bar{t}}_{ub_2} = 1$, this implies that at least one of the two integer solutions is feasible in constraint (16). □

The above result is used for finding an approximate solution to the pricing problem as follows. First, the linear programming relaxation of the pricing problem is solved. If the solution is not integer, we find the larger one of the two

values $r^{\bar{t}}_{ub_1}$ and $r^{\bar{t}}_{ub_2}$, and it determines which one of the two integer points is feasible in (16).

4.3 Upper bounding

The primary way to find feasible solutions to IP and upper bounds to its optimal value is to solve the integer version of the restricted master problem. As an additional means for finding upper bounds we employ a heuristic local search procedure. The starting point for this procedure is the schedules provided by the pricing problems, one for each UE, in one iteration.

These schedules are most likely to violate the constraint (3), which state that each subcarrier can be assigned to at most one UE at each time slot. The local search procedure gradually converts them in order to reduce subcarrier conflicts and eventually find a feasible solution and an upper bound to IP. In addition, the schedules that constitute the feasible solution are inserted as columns in the master problem. Thereby the master problem is supplied with more degrees of freedom, which can enable it to produce better upper bounds. In the local search, we have chosen to stop when the first feasible solution is found, but not to further improve the solution, since this would become too computationally expensive.

Let $\overline{x}^t_{ub}$ be a solution that is satisfying (2), (4), and (5) but not (3). Let $u = \overline{u}$ be a user such that $a_{nb}\overline{x}^t_{ub} = 1$ and $\sum_u \sum_b a_{nb}\overline{x}^t_{ub} > 1$ holds for some n and t, meaning that some subcarrier is assigned to both user $\overline{u}$ and another user at some time slot, which circumstance is defined as a conflict.

Update the schedule for user $\overline{u}$, by solving the problem

$$\text{minimize} \quad \sum_{b=0}^{B}\sum_{t=1}^{T} p_{\overline{u}b}x_{\overline{u}b}^{t} + M\sum_{n=1}^{N}\sum_{t=1}^{T} w_{nt} \tag{19}$$

$$\text{subject to} \quad \sum_{b=0}^{B}\sum_{t=1}^{T} r_{\overline{u}b}^{t}x_{\overline{u}b}^{t} \geq d_{\overline{u}} \tag{20}$$

$$\sum_{b=0}^{B} a_{nb}x_{\overline{u}b}^{t} - w_{nt} \leq 1 - \sum_{u\neq\overline{u}}\sum_{b=0}^{B} a_{nb}\overline{x}_{ub}^{t} \qquad \begin{matrix} n = 1,\ldots,N \\ t = 1,\ldots,T \end{matrix} \tag{21}$$

$$\sum_{b=0}^{B} x_{\overline{u}b}^{t} = 1 \qquad t = 1,\ldots,T \tag{22}$$

$$\sum_{b=0}^{B}\sum_{t=1}^{T}[(1-\overline{x}_{\overline{u}b}^{t})x_{\overline{u}b}^{t} + \overline{x}_{\overline{u}b}^{t}(1-x_{\overline{u}b}^{t})] \leq R \tag{23}$$

$$x_{\overline{u}b}^{t} \in \{0,1\} \qquad t = 1,\ldots,T, b = 0,\ldots,B, \tag{24}$$

where w_{nt} is an artificial variable and M is a large positive constant.

Constraint (23) defines a neighborhood that restricts the distance in 1-norm between the current solution $x_{\overline{u}b}^{t}$ and the updated schedule to be at most R. This approach is similar to the RINS heuristic, see Danna et al. (2005). The reason for bounding the distance using a 1-norm is that this restriction can be formulated using a single linear constraint. We use $R = 4$, which implies that the largest possible update in the schedule for user $\overline{u}$ is to replace a non-empty block in one time slot by a non-empty block in another time slot.

The updating procedure is initiated with the schedules found by the pricing problem (15)–(18). It is then repeated for users with a subcarrier conflict and it terminates when constraint (3) is fulfilled, that is, when a feasible solution and an upper bound to IP is found. Finally, the schedules found are inserted as auxiliary columns into the master problem, in order to accelerate the progress of finding strong upper bounds. To avoid any bias in the feasible solutions to the IP found by the local search procedure, the users are considered in a random order, which is used for only one application of the procedure. The summary of the algorithm is shown below.

```
Input: single user schedules produced by the pricing problem;
initialization;
creat permuted sequence of users;
while solution is infeasible do
    while sequence not finished do
        pick next user in sequence;
        if subcarrier conflict exists then
            solve problem (19)–(24);
            if solution improved then
                update solution;
            else
                if all users considered since last update then
                    break;
                end
            end
        end
    end
    restart sequence;
end
```

Algorithm 2: Local Search

4.4 Summary of enhanced column generation

The enhanced column generation is a combination of column generation, master problem stabilization, an initial phase of approximate column generation, and a late phase of improving upper bounds through the local search. Before presenting the algorithm, we first define the following notations: I = the number of initial iterations with approximate column generations, and $init_{ls}$ = the number of initial iterations before the local search is included. Then the algorithm is summarized as follows.

```
initialization;
set dual solution to zero;
solve all pricing problems, compute LBD and add columns to the RLMP;
set the size of the dual box;
for i = 1, ..., I do
    solve the SRLMP to obtain a dual solution;
    solve approximate pricing problems to get columns for the SRLMP;
    update box;
end
set dual solution to zero;
set the size of the dual box;
while (UBD − LBD)/LBD > ε do
    solve the SRLMP to obtain a dual solution;
    solve the RIMP to get UBD;
    solve all pricing problems and update LBD;
    if profitable columns are found then
        add profitable columns in the SRLMP;
    else
        if the box is not active then
            break;
        end
    end
    if iteration counter > init_ls then
        apply local search, update UBD and add columns to the RLMP;
    end
end
```

Algorithm 3: Enhanced Column Generation

5 Numerical results

We compare and evaluate the computational performance of both the standard and enhanced column generation methods, as well as that of the basic integer programming formulation. As evaluation environment we consider the uplink SC-FDMA of a single LTE-cell with randomly distributed UEs. Key simulation parameters are given in Table 1. The channel gain consists of path loss, shadowing, as well as multipath fading. The path loss follows the widely used COST 231 model that

extends the Okumura-Hata model for urban scenarios. By the COST 231 channel model, path loss is frequency dependent. Shadowing follows the log-normal distribution, for which 8 dB standard deviation is assumed. Multipath fading is described by the Rayleigh model.

Table 1: Simulation parameters

Parameter	**Value**
Cell radius	1000 m
Carrier frequency	2000 MHz
Subcarrier bandwidth	15 kHz
UE power constraint	200 mW
Subcarrier power constraint	10 mW
Power spectral density of noise	-174 dBm/Hz
Path loss	COST-231-HATA
Shadowing	Log-normal, 8 dB standard deviation
Multipath fading	Rayleigh fading

The characteristics of the used instances are described in Table 3, where "Ins." stands for instance and d_u is the demand for each UE. The user demands are drawn from uniform distributions over the intervals given in the table, and are in kilobits (kb) over the entire scheduling period, which consist of 5 to 50 time slots, each being 0.5 ms. Further, $rate$ is the corresponding bit rate in kilobits per second (kb/s), calculated as $2000 * d_u/T$. The considered value range of $rate$ is realistic with respect to real-life scenarios. The number of UEs, between 10 and 50, is representative for a cell radius of 1 km within an urban area. The users are located randomly and uniformly within the cell area.

Computations are carried out on a server equipped with two Intel Xeon 2.67 GHz CPUs, each with 6 cores. Each core has its own cache memory of size 12 MB and the machine as a whole has 72 GB of main memory. Typically each run uses 5–8 GB of main memory or similar. The IP model in Section 2 and the column generation schemes are implemented in AMPL (Fourer et al., 2002) and solved by CPLEX 12.5 (IBM ILOG CPLEX Optimizer, 2009). This solver can run on a single processor as well as on multi-core systems. In the latter case, parallel processing is available for solving integer programs with branch-and-bound. In our case, parallel processing is utilized while solving the integer programming elements, that is, the pricing problems, RIPM and in the local search.

In all our experiments, an accuracy (or gap), that is, a maximal deviation between the lower and upper bounds, of 1% is demanded. The limit for the real elapsed time is set to 1800 seconds for all instances, but a few instances are solved with a time limit of 3600 seconds since they are particularly difficult. For the column generation schemes, the time limit is only checked once per main iteration, and therefore the actual elapsed time can slightly exceed the time limit.

For the standard column generation approach, the problems RLPM and RIPM are both solved in each iteration. The enhanced column generation is composed of three phases. Some preliminary experiments and calibration of its parameters led to the following choices of strategies and parameter values. The phase of approximate column generation is designed to run for the first $I = 5$ iterations, while the phase of exact column generation and the solution of RIPM starts thereafter. The box radius δ is chosen to be 1.0 during the phase with approximate column generation and 0.5 during the phase with exact column generation.

The values of the algorithm parameters I and δ in the exact column generation phase were chosen empirically via a small set of computational experiments. This is illustrated in Table 2, which provides insights from varying δ and I for the eight instances 1.1-$2, 2.1$-$2, 3.1$-$2, 4.1$-2 in Table 3. In order to analyze how the values of these parameter affect the results, one of them is kept fixed while the other is varied. For each instance we find the value of the parameter in question that gives the shortest solution time and the smallest gap. These values are referred to as the winners for time and gap respectively. The table shows the number of winners for the various values of the parameters δ and I, respectively.

The local search heuristic can become time-consuming, and we therefore choose to postpone its inclusion for $init_{ls} = 10$ iterations. Further, each pricing problem is set with a time limit of 30 seconds to avoid the solution time to become too long in any iteration. If a pricing problem is not solved to optimality within 30 seconds, then the calculation of LBD uses the lower bound on the optimal value provided by CPLEX at termination.

Table 2: Parameter value justification through counting the number of winners for each value for eight instances. Here, win. = winners.

	δ					I				
parameter and values	0.1	0.25	0.5	1.0	2.5	1	3	5	7	9
# win. on time	1	1	6	0	0	0	1	7	0	0
# win. on gap	0	1	7	0	0	0	0	7	1	0

The comparison between using the formulation IP, the standard column generation (SCG) and the enhanced column generation (ECG) is shown in Table 4. The notation "n.a." refers to the case that no result is available when the program terminates because of lack of memory, while "n.i." means that no feasible integer solution is found within the time limit. From the results presented in this table, it is evident that the straightforward integer programming formulation IP performs very poorly, compared to the column-oriented formulation. The IP formulation is undoubtedly inefficient, both with respect to the computing time required for finding a proven near-optimal solution and to the use of computer memory. As can be seen, the LP values obtained are low, meaning that the linear programming relaxation of this formulation is weak, which causes the final lower bounds to be poor. The relative deviation between the bounds is dramatically improved using the column-oriented formulation and column generation approaches, as compared to the use of the formulation IP. Especially the lower bounds are significantly improved, which means that the column-oriented model constitutes a much stronger formulation than IP. As was mentioned in Section 3.2, due to the lack of integrality property, the lower bound given by the column-oriented model could be strictly better than that from the LP relaxation of the first formulation, even though the difference in the strengths of the two formulations can only be observed via numerical results.

We also make the following observations from the results of the table. Firstly, it can be seen that the ECG successfully provides a much better accuracy than the SCG, although the accuracy reached by the SCG already provides a very tight bounding interval for the global optimality, compared to the formulation IP. Secondly, for most of the instances the UBDs are substantially improved by ECG while the LBDs are comparable, and it can be observed that the SCG method tends to have difficulties finding integer feasible solutions or good integer feasible solutions for instances that are tighter and have larger variations in the users' rate demands. This verifies the intended enhanced upper bounding quality of ECG. Thirdly, the time spent by ECG is mostly shorter than or similar to the time spent by SCG, while the time spent by both SCG and ECG to reach an accuracy of 1% is mostly much less than the given time limit. Lastly, from the results we observe that the performance gap between solving the program IP directly and the proposed column generation methods becomes smaller while T decreases, because of reduced problem size, yet our method remains superior (both in time and accuracy).

Table 5 shows the arithmetic average time consumed per iteration of the different components of both the standard and the enhanced column generation for the

larger instances shown in the previous table. For both methods, RLPM and RIPM consume very little time. For the standard column generation, the time needed in the column generation step dominates entirely. In the enhanced column generation, the approximate column generation costs much less than the exact one. The local search heuristic is not frequently needed to reach a high-quality UBD, but when it is, it can become time-consuming. With respect to the number of iterations, the enhanced column generation is not always superior to the standard column generation; however, the average time per iteration of the former is mostly less than or similar to that of the latter. We observe that the average number of iterations over the instances performed by SCG is 9.2, while it for ECG is 10.8. This implies that ECG performs on average only 5.8 iterations with exact pricing, thanks to contributions from the iterations with approximate column generation and the stabilization.

For instance no. 1.1, which is rather difficult, the convergence of the SCG and ECG is illustrated in Fig. 1 with iteration number on the horizontal axis and in Fig. 2 with time on the horizontal axis. As can be seen from Fig. 1, the ECG shows a steadily improving lower bound, which is an expected effect from the introduction of the stabilization of the master problem. In contrast, the lower bound in the SCG is not improving at all. In the SCG, the problem RIPM is infeasible in the first 5 iterations, while in the ECG, the problem RIPM is not solved in the first 5 iterations, as mentioned earlier. Note that for the ECG the upper bound drops rather quickly, which is thanks to improved quality of the generated columns, and we believe this to be an effect of the stabilization. Further, the upper bound improves especially after the initialization of the local search after 10 iterations. We can see from Fig. 2 that, with the functioning of the approximate column generation, the bounds close more rapidly in the ECG than in the SCG. Note also that the accuracy eventually reached with the ECG is much higher than from the SCG.

We also made experiments to examine the stability of the three solution approaches with fast fading. For fixed user locations and demands, fast fading has been simulated by generating multiple samples from the Rayleigh distribution to get different rate data as input. For the smallest problem size, 100 samples were generated, while for the other sizes 10 samples were generated. The results are shown in Table 6. The table shows arithmetic means over the samples of each size, and based on these average results, the proposed method is robust in exhibiting a superior performance. For size $20 \times 30 \times 40$, four instances are excluded from the average values since no UBDs are found by SCG within the time limit. However, ECG always finds feasible solutions, which further verifies its stronger upper bounding capability. In addition, for the problem size $20 \times 20 \times 40$, we

use Fig. 3 to provide details of the distribution of solution time as well as accuracy. Note that even though the absolute gap values are all small, the differences between the solution approaches are significant. The other problem sizes show similar patterns.

As the final part of experiments, we study the effect of user demand on the performance of the three approaches. To this end, we considered an instance of size $40 \times 30 \times 40$, and constructed various combinations of user demands of 64, 128, and 512 kb/s, such that the total demand is approximately the same. All other data are kept fixed. The distributions of the numbers of users with the three demands are shown in Table 7. It can be seen that the relative performance of the three approaches is not much influenced by the variation in the distribution of the demands. In some cases, SCG did not find any feasible solution within the time limit, while in the other cases the UBDs are not near-optimal. The ECG approach is however always capable of finding near-optimal feasible solutions. These results are in line with the earlier observation that SCG has difficulties finding good feasible solutions when the demands have larger variations.

To summarize the effects of the algorithmic components introduced in the ECG, we first comment that the behavior of the lower bound shown in Fig. 1, with a steadily improving quality, is due to the stabilization and typical for all the test runs with ECG. Also thanks to the stabilization, better dual information is provided to the pricing problem, which then in turn tends to find columns that are advantageous in RIPM for reaching good upper bounds. Further, the approximate pricing phase acts as an initialization which is computationally very cheap but still contributes substantially in reducing the necessary number of iterations with exact pricing. Finally, the local search heuristic often strengthens the upper bounding significantly, either directly by finding high-quality feasible solutions, or indirectly by supplying advantageous UE schedules to the RIPM, so that solving this problem can produce a high-quality upper bound in a later iteration. The heuristic can therefore be seen as a safeguard, that facilitates the finding of good upper bounds.

Table 3: Instance characteristics. Here, Ins. = instance

Ins.	$U \times N \times T$	B	d_u (kb)	$rate$ (kb/s)	Ins.	$U \times N \times T$	B	d_u (kb)	$rate$ (kb/s)
a.1	$10 \times 15 \times 5$	36	0.8	320	2.1	$30 \times 60 \times 40$	1830	6–9	300-450
a.2	$10 \times 15 \times 5$	36	0.1–2	80–800	2.2	$30 \times 60 \times 40$	1830	7–8	350-400
b.1	$10 \times 20 \times 10$	210	2	400	2.3	$30 \times 60 \times 40$	1830	4–12	200-600
b.2	$10 \times 20 \times 10$	210	0.1–3	60–600	3.1	$40 \times 60 \times 40$	1830	5–8	250-400
c.1	$20 \times 30 \times 20$	465	3	300	3.2	$40 \times 60 \times 40$	1830	6–7	300-350
c.2	$20 \times 30 \times 20$	465	0.1–4.5	45–450	3.3	$40 \times 60 \times 40$	1830	3–10	150-500
d.1	$30 \times 40 \times 30$	820	4	267	4.1	$50 \times 60 \times 40$	1830	4–7	200-350
d.2	$30 \times 40 \times 30$	820	0.1–5	33–333	4.2	$50 \times 60 \times 40$	1830	5–6	250-300
e.1	$40 \times 50 \times 30$	1275	4	267	4.3	$50 \times 60 \times 40$	1830	3–8	150-400
e.2	$40 \times 50 \times 30$	1275	0.1–6	40–400	5	$20 \times 80 \times 50$	3240	8–14	320-560
1.1	$20 \times 60 \times 40$	1830	6–12	300–600	6	$30 \times 80 \times 50$	3240	5–12	200-480
1.2	$20 \times 60 \times 40$	1830	10–11	500–550	7	$40 \times 80 \times 50$	3240	5–11	200-440
1.3	$20 \times 60 \times 40$	1830	10–15	500–750	8	$50 \times 80 \times 50$	3240	5–10	200-400

Table 4: Comparison of UBD, LBD and accuracy obtained by IP, SCG and ECG. Here, LP is the optimal value of the linear programming relaxation of IP, acc.= accuracy = (UBD-LBD)/LBD, n.a. = not available because of lack of memory, n.i. = no integer feasible solution found within the time limit. The winners with respect to accuracy are marked in boldface.

	IP					SCG				ECG			
Ins.	LP	time	LBD	UBD	acc. (%)	time	LBD	UBD	acc. (%)	time	LBD	UBD	acc. (%)
a.1	490	57	550	550	0.00	14	550	570	3.64	1	550	550	**0.00**
a.2	559	1803	608	620	1.97	244	620	n.i.	n.i.	42	620	620	**0.00**
b.1	1424	1803	1450	1470	1.38	1294	1457	1470	0.91	1489	1464	1470	**0.44**
b.2	1529	1803	1549	1570	1.34	1803	1551	n.i.	n.i.	887	1558	1570	**0.79**
c.1	2437	1809	3327	3410	2.49	1806	3369	3420	1.50	1234	3389	3410	**0.63**
c.2	2066	1809	2645	2720	2.85	1894	2677	2710	1.23	1407	2694	2720	**0.95**
d.1	3658	1270	5708	5740	0.56	357	5729	5750	0.37	173	5729	5740	**0.20**
d.2	2442	1831	3993	4120	3.18	1274	4096	4130	0.84	1456	4096	4120	**0.60**
e.1	4128	1814	7568	7650	1.08	478	7607	7680	0.95	637	7620	7660	**0.53**
e.2	3660	1847	6391	6490	1.55	1979	6475	6620	2.24	2123	6475	6500	**0.39**
1.1	3728	1810	4375	4730	8.12	1932	4451	4730	6.26	1801	4579	4700	**2.64**
1.2	4342	1896	4684	6030	28.75	1890	4618	5370	16.28	1865	5046	5270	**4.43**
1.3	4916	3630	5114	5950	16.35	3779	5329	6050	13.52	3755	5690	5880	**3.35**
2.1	4849	1842	6062	6230	2.77	910	6090	6130	0.66	919	6090	6110	**0.33**
2.2	4676	n.a.	n.a.	n.a.	n.a.	454	6030	6060	0.50	366	6036	6050	**0.23**
2.3	4956	1858	6506	n.a.	n.a.	1910	5857	n.i.	n.i.	1869	6858	7260	**5.86**
3.1	5076	1882	8001	8230	2.86	889	8057	8130	0.90	437	8055	8060	**0.02**
3.2	5168	1851	8026	n.i.	n.i.	505	8050	8110	0.74	464	8050	8050	**0.00**
3.3	5632	1834	8157	8630	5.80	1953	8172	n.i.	n.i.	2047	8183	8410	**2.77**
4.1	5885	1833	n.i.	n.i.	n.i.	961	9940	10020	0.80	1177	9940	9940	**0.00**
4.2	5819	n.a.	n.a.	n.a.	n.a.	731	9990	10060	0.70	492	9990	9990	**0.00**
4.3	5757	1818	9748	n.i.	n.i.	1877	9669	9790	3.06	1948	9789	10000	**2.16**
5	3334	1834	4138	4490	8.51	1828	4229	4320	2.15	1820	4230	4320	**2.12**
6	n.a.	n.a.	n.a.	n.a.	n.a.	455	6260	6300	0.64	668	6260	6260	**0.00**
7	n.a.	n.a.	n.a.	n.a.	n.a.	885	8000	8030	0.37	484	8000	8000	**0.00**
8	n.a.	n.a.	n.a.	n.a.	n.a.	615	10000	10010	0.10	601	10000	10000	**0.00**

Table 5: Arithmetic average time per column generation iteration in SCG and ECG. Here, iter.= iterations, App. = approximate, Ex.= exact, LS = local search

	SCG					ECG						
Ins.	# iter.	total	RLPM	CG	RIPM	# iter.	total	RLPM	App. CG	Ex. CG	LS	RIPM
1.1	12	161.98	0.04	160.89	0.05	16	121.38	0.06	12.35	139.71	56.93	0.09
1.2	15	264.78	0.09	264.55	0.14	11	208.98	0.04	13.43	320.89	305.24	0.11
1.3	14	269.93	0.10	269.71	0.13	15	250.31	0.06	19.49	299.51	132.02	0.11
2.1	9	101.19	0.06	101.06	0.08	12	49.18	0.08	19.25	62.63	26.84	0.12
2.2	6	75.67	0.05	75.58	0.04	10	36.65	0.06	20.52	52.58	0.00	0.08
2.3	4	477.52	0.63	475.42	1.48	11	169.88	0.13	25.34	260.20	179.38	0.13
3.1	7	126.93	0.07	126.57	0.29	8	54.56	0.06	23.66	105.80	0.00	0.12
3.2	6	84.13	0.06	83.99	0.08	10	46.41	0.06	25.13	67.17	0.00	0.38
3.3	11	177.56	0.22	176.28	1.06	12	170.55	0.12	38.85	185.88	273.78	0.30
4.1	8	120.18	0.12	119.60	0.46	12	98.11	0.12	14.60	71.77	68.41	0.37
4.2	8	91.37	0.10	91.07	0.20	10	49.20	0.09	28.44	69.56	0.00	0.00
4.3	20	93.87	0.28	92.67	0.92	12	162.32	0.16	34.52	189.39	216.34	2.14
5	16	114.2	0.03	114.16	0.04	14	130.00	0.09	31.55	168.48	35.90	0.11
6	3	151.81	0.05	151.47	0.29	8	83.45	0.12	45.28	145.33	0.00	1.48
7	5	177.06	0.07	176.92	0.07	6	80.72	0.10	57.22	197.49	0.00	0.14
8	3	204.95	0.08	204.81	0.06	6	100.14	0.12	70.33	248.31	0.00	0.18

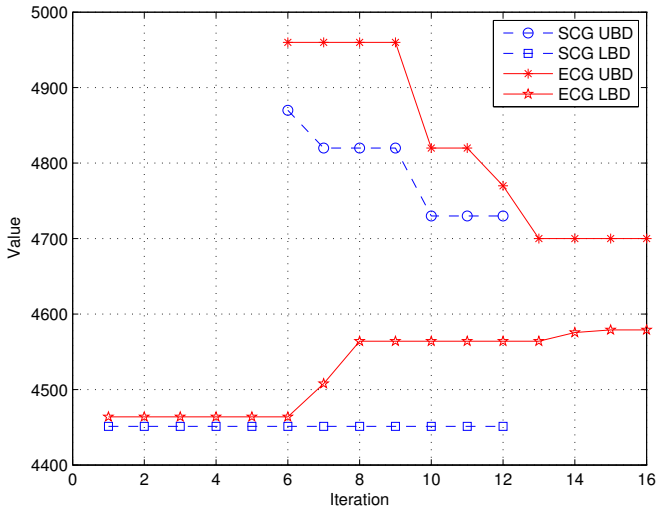

Figure 1: Comparison of convergence for SCG and ECG with respect to the number of iterations

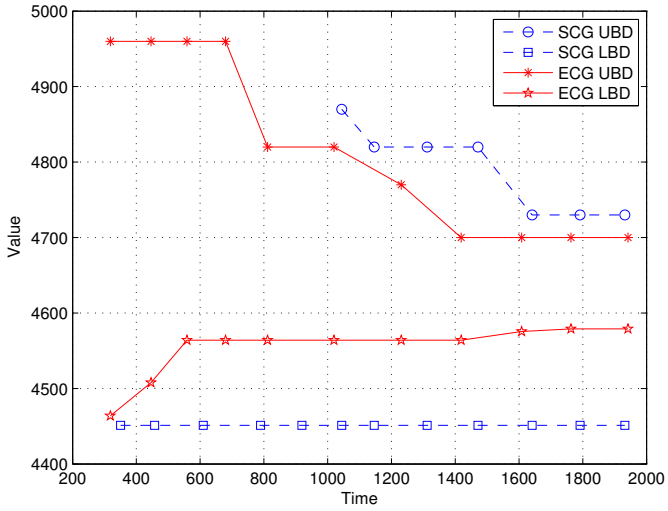

Figure 2: Comparison of convergence for SCG and ECG with respect to the solution time

Table 6: Comparison of UBD, LBD and accuracy obtained by IP, SCG and ECG for fast fading.

	IP					SCG				ECG			
Size	LP	time	LBD	UBD	acc.(%)	time	LBD	UBD	acc.(%)	time	LBD	UBD	acc.(%)
$10 \times 20 \times 5$	460	1615	509	521	2.36	37	521	527	1.19	10	524	527	**0.62**
$20 \times 20 \times 40$	2084	1816	2107	2181	3.51	916	2181	2194	0.60	266	2181	2186	**0.23**
$20 \times 30 \times 40$	2776	1822	2917	2975	1.99	1791	2968	3042	2.45	1456	2968	3010	**1.40**
$20 \times 40 \times 40$	2836	1526	3288	3327	1.19	618	3335	3373	1.14	532	3335	3355	**0.61**
$20 \times 50 \times 40$	2971	1831	3311	3617	9.24	1222	3387	3416	0.86	964	3391	3405	**0.43**
$20 \times 60 \times 40$	2949	1837	3576	3845	7.52	1450	3662	3693	0.84	1118	3663	3680	**0.47**

Table 7: Comparison of UBD, LBD and accuracy obtained by IP, SCG and ECG with fixed total demand. Here, Distribution refers to the order 64, 128, and 528 kb/s.

	IP					SCG				ECG			
Distribution	LP	time	LBD	UBD	acc.(%)	time	LBD	UBD	acc.(%)	time	LBD	UBD	acc.(%)
$5-33-2$	2427	1808	3136	3350	6.8	1955	3275	n.i.	n.i.	1950	3292	3360	**2.1**
$11-26-3$	2358	1801	3076	3290	7.0	1893	3232	3660	13.2	1944	3228	3360	**4.1**
$17-19-4$	2326	1803	3008	3220	7.1	1843	3173	3490	10.0	1825	3170	3240	**2.2**
$23-12-5$	2311	1802	2938	3120	6.2	1860	3079	3390	10.1	1836	3079	3140	**2.0**
$29-5-6$	2308	1802	2845	3020	6.2	1850	2987	3320	11.2	1851	2983	3050	**2.2**

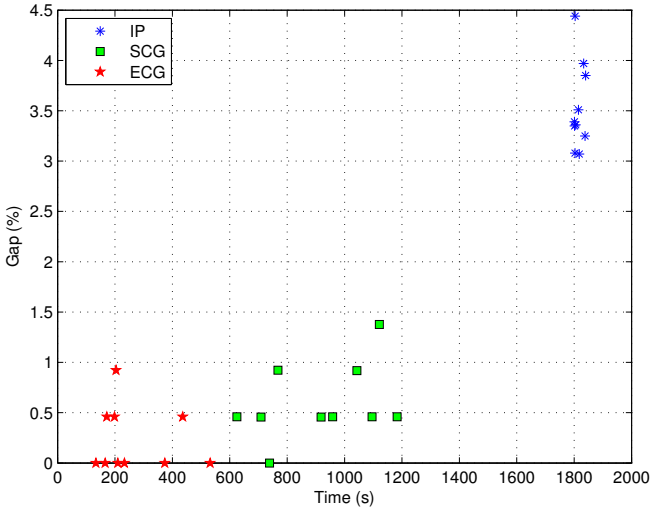

Figure 3: Distribution of solutions from IP, SCG and ECG with respect to solution time and gap

6 Conclusion and future work

We have studied the problem of power efficient scheduling in uplink SC-FDMA systems with localized subcarrier allocation. This is an NP-hard problem, and previously proposed algorithms are mainly greedy-based heuristics. We are aiming to deliver tight bounds on the global optimum as an effective means for benchmarking the performance of suboptimal algorithms.

We developed a basic integer linear programming formulation, a column-oriented formulation, and both standard and enhanced column generation solution schemes for this problem. The basic integer programming formulation is computationally demanding with standard software, because of the weakness of its linear programming relaxation. In contrast, the column-oriented model provides a strong formulation of the problem. Further, the heuristic principle for finding integer feasible solutions that is embedded in the enhanced column generation scheme provides high-quality upper bounds.

The overall conclusion of our numerical evaluation is that the column-oriented approaches outperform the integer linear programming formulation with respect to both the quality of the bounding interval and solution time. Moreover, although

the standard column generation already yields significantly improved results compared to the integer programming formulation, the enhanced column generation successfully further sharpens the bounding interval using comparable or even less time than the standard column generation. This is due to the improved convergence behaviour that results from the stabilization, the cheap approximate column generation, and the enhanced upper bounding provided by the local search heuristic; the two latter enhancements are tailored to the problem structure.

The successful results presented validate our motive for approaching this scheduling problem by column generation, which is the observation that a key success factor for the solution of computationally demanding discrete optimization models is the use of strong formulations, and that one option for constructing strong formulations is the use of column-oriented models and column generation.

For future work, we conclude with some suggestions for extensions of the current study. The objective in the paper has been on energy minimization, which is a very active topic in the engineering field. We remark that the underlying idea of problem-solving by means of column generation extends to other, related problem setups. We first note that, in addition to power minimization, it is of significance to consider other performance objectives, such as to maximize rates subject to power limits, a weighted combination of power and rate, user-specific objective functions, as well as an objective that reflects priorities between the users. These extensions allow for considering differentiation between users. For example, while for some users it is of importance to save power, for others the achieved rate has higher preference. Another extension is to consider mixed voice and date users, having different time scales in scheduling in terms of the amounts of demand. The enhanced column generation method can still be applied without significant modifications, and such an extension may be of interest for the engineering community.

In addition, services with different delay latencies can be handled in a rolling horizon manner, where services with low delay latency are given high priority within the current scheduling horizon, by means of fixed rate demands, while services with high delay latency are scheduled through rate maximization within the remaining rate capacity.

We conclude by pointing out that some of the solution techniques used can be extended to other applications of column generation. Particularly, our technique of combining column generation with a heuristic that converts columns into an integer feasible solution, with the aim of upper bounding and the generation of additional columns to the master problem, remains applicable for problems for which column generation is frequently deployed, for example bin packing and

parallel machine scheduling.

Acknowledgments

The work of Yixin Zhao has been supported by the Research School in Interdisciplinary Mathematics at Linköping University. The work of Di Yuan has been supported by the Excellence Center at Linköping - Lund in Information Technology, Centrum för Industriell Informationsteknologi, Linköping University, EC FP7 Marie Curie Project 318992. The work of Lei Lei has been supported by the Chinese Scholarship Council. We thank all the reviewers for their remarks and suggestions that have enabled us to improve the content and clarity of the paper.

References

Ahmad A, Assaad M (2011) Polynomial-complexity optimal resource allocation framework for uplink SC-FDMA systems. In: Proceedings of IEEE GLOBECOM conference, Houston, Texas, USA, 5–9 December, pp 1–5

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Operations Research 46(3):316–329

Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear Programming: Theory and Algorithms. John Wiley and Sons, New York, NY

Bertsimas D, Tsitsiklis JN (1997) Introduction to Linear Optimization. Athena Scientific and Dynamic Ideas, Belmont, MA

Cheong YW, Cheng RS, Lataief KB (1999) A multiuser OFDM with adaptive subcarrier, bit, and power allocation. IEEE Journal on Selected Areas in Communications 17(10):1747–1758

Danna E, Rothberg E, Le Pape C (2005) Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming 1(102):71–90

Dantzig GB, Wolfe P (1960) Decomposition principles for linear programs. Operations Research 8(1):101–111

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball MO, et al. (eds), Network Routing, Handbooks in Operations Research and Management, Vol. 8, North-Holland, Amsterdam, pp 35–139

du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. Discrete Mathematics 194(1–3):229–237

European Telecommunications Standards Institute (2009) Evolved universal terrestrial radio access (E-UTRA): User Equipment (UE) radio transmission and reception. 3GPP Technical Specification 36.101.version 8.4.0. Release 8. http://www.3gpp.org. Accessed January 2013

Fourer R, Gay DM, Kernighan BW (2002) AMPL: A Modeling Language for Mathematical Programming, Duxbury Press, Pacific Grove, CA

Gao N, Wang X (2011) Optimal subcarrier-chunk scheduling for wireless OFDMA systems. IEEE Transactions on Wireless Communications 10(7):2116–2118

Hiriart-Urruty JB, Lemaréchal C (1993) Convex Analysis and Minimization Algorithms, Vol. 2, Advanced Theory and Bundle Methods. Springer-Verlag, Berlin

IBM ILOG CPLEX Optimizer (2009) http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. Accessed February 2012

Kim D, Kim J, Kim H (2010) An efficient scheduler for uplink single carrier FDMA system. In: Proceedings of the 21st IEEE PIMRC, Istanbul, Turkey, 26–29 September, pp 1348-1353

Larsson T, Patriksson M, Rydergren C (2004) A column generation procedure for the side constrained traffic equilibrium problem. Transportation Research Part B: Methodological 38(1):17–38

Lee S-B, Pefkianakis I, Meyerson A, Xu S, Lu S (2009) Proportional fair frequency-domain packet scheduling for 3GPP LTE uplink. In: Proceedings of the 28th IEEE INFOCOM conference, Rio de Janeiro, Brazil, 19–25 April, pp 2611–2615

Lei L, Fowler S, Yuan D (2013a) Improved resource allocation algorithm based on partial solution estimation for SC-FDMA systems. In: Proceedings of the IEEE 78th Vehicular Technology Conference (VTC), Las Vegas, Nevada, USA, 2–5 September, pp 1–5

Lei L, Yuan D, Ho KC, Sun SM (2013b) A unified graph labeling algorithm for consecutive-block channel allocation in SC-FDMA. IEEE Transactions on Wireless Communications 12(11):5767–5779

Lei L, Angelakis V, Yuan D (2012) Performance analysis of chunk-based resource allocation in wireless OFDMA systems. In: Proceedings of the IEEE 17th CAMAD conference, Berlin, Germany, 17–19 September, pp 90–94

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. Operations Research 53(6):1007–1023

Marsten RE, Hogan WW, Blankenship JW (1975) The boxstep method for large-scale optimization. Operations Research 23(3):389–405

Miao G, Himayat N, Li GY (2010) Energy-efficient link adaptation in frequency-selective channels. IEEE Transactions on Communications 58(2):545–554

Myung HG, Lim J, Goodman DJ (2006) Single carrier FDMA for uplink wireless transmission. IEEE Vehicular Technology Magazine 1(3):30–38

Pisinger D (1995) A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operational Research 83(2):394–410

Safa H, Tohme K (2012) LTE uplink scheduling algorithms: performance and challenges. In: Proceedings of the 19th IEEE International Conference on Telecommunications, Regency Palace Hotel Adma Jounieh, Lebanon, 23–25 April, pp 1–6

Tse D, Viswanath P (2005) Fundamentals of Wireless Communication. Cambridge University Press, New York, NY

Vanderbeck F, Desaulniers G, Desrosiers J, Solomon MM (eds) (2005) Column Generation. Springer, New York, NY

Westerlund A, Göthe-Lundgren M, Larsson T (2006) A stabilized column generation scheme for the travelling salesman subtour problem. Discrete Applied Mathematics 154(15):2212–2238

Wilhelm WE (1995) A technical review of column generation in integer programming. Optimization and Engineering 2(2):159–200

Wolsey LA (1998) Integer Programming. John Wiley and Sons, New York, NY, pp 188–189

Wong IC, Oteri O, Mccoy W (2009) Optimal resource allocation in uplink SC-FDMA systems. IEEE Transactions on Wireless Communications 8(5):2161–2162

Zhao H, Larsson T, Yuan D, Rönnberg E, Lei L (2013) Power efficient uplink scheduling in SC-FDMA: Bounding global optimality by column generation In: Proceedings of IEEE 18th International Workshop on CAMAD, 2013 September 25–27, Berlin, pp 119–123