An Exact Algorithm for a Resource Allocation Problem in Mobile Wireless Communications

Adam N. Letchford* Qiang Ni† Zhaoyu Zhong*

To appear in *Computational Optimization & Applications*

Abstract

We consider a challenging resource allocation problem arising in mobile wireless communications. The goal is to allocate the available channels and power in a so-called *OFDMA system*, in order to maximise the transmission rate, subject to quality of service (QoS) constraints. Standard MINLP software struggled to solve even small instances of this problem. Using outer approximation, perspective cuts and several implementation "tricks", we are able to solve realistic instances in about one minute. A novel ingredient of our algorithm is what we call *pre-emptive cut generation*: the generation of cutting planes that are not violated in the current iteration, but are likely to be violated in subsequent iterations.

Keywords: mixed-integer nonlinear programming, mobile wireless communications, OFDMA systems

1 Introduction

Telecommunications provides a rich source of interesting, and often challenging, optimisation problems (see, e.g., Resende & Pardalos [22]). This paper is concerned with a mixed-integer convex optimisation problem that arises in *mobile wireless* communications systems. In such systems, mobile devices (such as smartphones or tablets) communicate with one another via transceivers called *base stations*. Each base station must periodically allocate its available resources (time, power and bandwidth) in order to receive and transmit data in an efficient way (see, e.g., Fazel & Kaiser [7]).

The problem under consideration arises when the base stations have an *Orthogonal Frequency-Division Multiple Access* (OFDMA) architecture. This means that the base station divides the available bandwidth into a

*Department of Management Science, Lancaster University, Lancaster LA1 4YX. E-mail: {A.N.Letchford,z.zhong1}@lancaster.ac.uk

†School of Computing and Communications, Lancaster University, Lancaster LA1 4WA. E-mail: q.ni@lancaster.ac.uk

number of frequency bands called *subcarriers*. Each subcarrier can be assigned to only one mobile device (or *user*) in any given time period, but a given user may be assigned to more than one subcarrier. The data transmission rate for any given subcarrier is a nonlinear function of the power allocated to that subcarrier.

There are several distinct optimisation problems associated with OFDMA (and related) systems, with differing objective functions, side-constraints and so on (e.g., [12,17–19,23,24,26,27,29,31–35]). The problem that we focus on in this paper is that of simultaneously allocating subcarriers to users and power to subcarriers, subject to certain quality of service (QoS) constraints called *rate constraints*, in order to maximise the total data transmission rate. We call this problem the *subcarrier and power allocation problem with rate constraints* (SPARC).

The SPARC has a natural formulation as a convex *mixed-integer nonlinear program* (MINLP). Since we found that standard software for convex MINLP was unable to solve even small instances of our problem in reasonable computing times, we developed our own specialised exact algorithm. It uses a combination of outer approximation [8] and perspective cuts [11,13], together with three implementation "tricks" of our own, which improve the running time by several orders of magnitude.

A novel ingredient of our algorithm, which turned out to be crucial, is what we call *pre-emptive cut generation*. By this, we mean the generation of cutting planes that are not violated in the current iteration, but are likely to be violated in subsequent iterations.

It turns out that our exact algorithm is capable of solving SPARC instances of realistic size and complexity to proven optimality in about one minute. In fact, instances with relatively loose QoS constraints can be solved in a fraction of a second. As far as we know, our algorithm is the first viable exact algorithm for a realistic OFDMA optimisation problem. It is also the first algorithm to apply perspective cuts to a problem in mobile wireless communications.

The paper is structured as follows. The relevant literature is reviewed in Section 2. In Section 3, we define the SPARC formally and present a convex MINLP formulation for it. The exact algorithm is described in Section 4. The results of some extensive computational experiments are given in Section 5, and some concluding remarks are made in Section 6.

We assume throughout that the reader is familiar with basic concepts of MINLP, such as continuous relaxation, convexity, lower and upper bounds, and branching. For tutorials, see, e.g., [1,5,30].

2 Literature Review

Now we briefly review the literature on optimisation in OFDMA and related systems. Good reference texts are [4, 7, 14].

2.1 Single-user systems

First, consider a single communications channel and a single user. The classical *Shannon–Hartley theorem* [25] states that the maximum data rate (in bits per second) that can be transmitted from a single channel is:

$$B \log_2 (1 + S/N),$$

where B is the bandwidth of the channel in Hertz, S is the average received signal power in watts, and N is the average noise power in watts. The quantity S/N is called the *signal-to-noise ratio*.

Now suppose that we still have a single user, but we now have a set I of subcarriers, and each subcarrier $i \in I$ has its own bandwidth B_i and noise power N_i. If we allocate p_i watts of power to subcarrier i, the data rate for that subcarrier will be $B_i \log_2(1 + p_i/N_i)$, which we denote by $f_i(p_i)$. A natural optimisation problem is then to maximise the total data rate subject to an overall power limit P. This can be formulated as the following NLP:

$$\max \left\{ \sum_{i \in I} f_i(p_i) : \sum_{i \in I} p_i \leq P,\ p \in \mathbb{R}_+^{|I|} \right\}. \tag{1}$$

Since the functions $f_i(p_i)$ are concave, this NLP can be solved efficiently by any standard technique for convex optimisation (see, e.g., Boyd & Vandenberghe [3]). It can also be solved quickly by a specialised iterative technique called *water filling*; see, e.g., [4, 14].

2.2 Multi-user systems

As mentioned in the introduction, OFDMA systems are multi-user systems, and each subcarrier can be assigned to an arbitrary user. If we let J denote the set of users, and $S_j \subset I$ denote the set of subcarriers allocated to user $j \in J$, then the total data rate for that user will be $\sum_{i \in S_j} f_i(p_i)$, and the total data rate for the system will be $\sum_{j \in J} \sum_{i \in S_j} f_i(p_i)$. One then faces optimisation problems in which one must simultaneously distribute the available power between the subcarriers, and allocate each subcarrier to a user, in order to meet some objective.

There are many papers on optimisation problems of this type. Wong & Cheng [31] minimise total power subject to individual quality of service (QoS) constraints that impose a lower bound on the data rate for each user. (We will call them *rate constraints*.) Rhee & Cioffi [23] achieve QoS in a

different way, by maximising the minimum data rate over all users, subject to a limit on total power. Kim *et al.* [17] consider the problem of maximising total data rate subject to a total power limit. Shen *et al.* [26] add a global QoS constraint to that problem. Seung *et al.* [24] enforce QoS by giving each user a weight, and maximising a weighted sum of the data rates. Yu & Lui [34] consider an extension of the problem in [24], in which there is interference between channels. Tao *et al.* [27] take the problem in [17], add rate constraints, and also consider an extension in which transmissions can suffer delays.

More recently, perhaps driven by environmental considerations, authors have focused on maximising *energy efficiency*, which is defined as total data rate divided by total power (e.g., [12, 15, 29, 32, 33, 35]).

It is proved in [18, 19] that many of the problem variants considered in the above papers are $\mathcal{NP}$-hard in the strong sense. Accordingly, in all of the above-mentioned papers, the authors use relaxation techniques to compute bounds, and heuristics to find feasible solutions. In this paper, we focus on exact methods.

3 Problem Definition and MINLP Formulations

We now formally describe the problem under consideration and give two MINLP formulations of it.

3.1 Problem definition

Although the methods developed in this paper can be applied to several OFDMA optimisation problems, we restrict attention to one specific problem, for the sake of brevity and clarity. As mentioned in the introduction, we call this problem the SPARC. A SPARC instance is given by sets I and J, a bandwidth $B_i > 0$ and noise $N_i > 0$ for each $i \in I$, a *user rate* $\ell_j \geq 0$ for each $j \in J$, and a power limit $P > 0$. As in [17, 26], the task is to allocate subcarriers to users, and power to subcarriers, in order to maximise the total data rate, subject to a constraint stating that the total power must not exceed P. In addition, however, we have rate constraints, as in [27, 31], stating that the total data rate for user j must be at least ℓ_j. We conjecture that the SPARC is $\mathcal{NP}$-hard in the strong sense.

3.2 Initial convex MINLP formulation

We formulate the SPARC as an MINLP as follows. For all $i \in I$ and $j \in J$, let x_{ij} be a binary variable, taking the value 1 if and only if user j is assigned to subcarrier i. Also let p_{ij} be a continuous variable, taking the value zero if $x_{ij} = 0$, but otherwise representing the amount of power supplied to

subcarrier i. We then have:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} f_i(p_{ij}) && && (2)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P && && (3)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 && (\forall i \in I) && (4)\\
& \textstyle\sum_{i\in I} f_i(p_{ij}) \ge \ell_j && (\forall j \in J) && (5)\\
& p_{ij} \le P x_{ij} && (\forall i \in I,\, j \in J) && (6)\\
& p_{ij} \in \mathbb{R}_+ && (\forall i \in I,\, j \in J) && (7)\\
& x_{ij} \in \{0,1\} && (\forall i \in I,\, j \in J). && (8)
\end{aligned}$$

The objective function (2) represents the total data rate. The constraint (3) imposes the power limit. The constraints (4) ensure that each subcarrier is allocated to at most one user. The constraints (5) are the user rate constraints. The constraints (6), called *variable upper bounds* (VUBs), ensure that p_{ij} is zero whenever x_{ij} is zero. The remaining constraints are the usual non-negativity and binary conditions.

Note that, for all $i \in I$, the function f_i is concave over the domain $[0, P]$. As a result, the objective function (2) is concave, and the constraints (5) are convex. This means that the MINLP is convex, and therefore its continuous relaxation can be solved efficiently via convex programming techniques.

Many other OFDMA optimisation problems can be formulated in a similar way. For brevity, we give just three examples. If one wishes to give each user $j \in J$ a weight $w_j \ge 0$, as in [24, 34], then one changes the objective function (2) to $\sum_{j\in J} w_j \sum_{i\in I} f_i(p_{ij})$. If one wishes to impose an upper bound u on the power assigned to each subcarrier, as in [15], one changes P to u in the VUBs (6). If one does not have enough capacity to satisfy all of the users, and one wishes to maximise the number of satisfied users, one changes the right-hand side of the rate constraints (5) to $\ell_j z_j$, where z_j is a new binary variable, and changes the objective function to $\sum_{j\in J} z_j$.

3.3 Modified convex MINLP formulation

It is well known (see, e.g., Section 2 of [28]) that MINLPs can often be made easier to solve by the addition of new variables, representing simple components of nonlinear functions. This turned out to be the case for our problem. Accordingly, for $i \in I$ and $j \in J$, we introduce a new non-negative continuous variable, say r_{ij}, representing the quantity $f_i(p_{ij})$. (We use r_{ij} because it represents the data *rate* of subcarrier i when $x_{ij} = 1$.) We then modify the formulation to:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} r_{ij} && && (9)\\
\text{s.t.} \quad & (3), (4), (6) - (8) && && (10)\\
& \textstyle\sum_{i\in I} r_{ij} \ge \ell_j && (\forall j \in J) && (11)
\end{aligned}$$

$$r_{ij} \leq f_i(p_{ij}) \qquad (\forall i \in I, j \in J). \tag{12}$$

$$r_{ij} \in \mathbb{R}_+ \qquad (\forall i \in I, j \in J). \tag{13}$$

With these modifications, eveything is linear apart from the (convex) constraints (12).

4 An Exact Algorithm for the SPARC

We now present our exact algorithm for the SPARC. Subsection 4.1 presents the algorithm in its simplest form. Enhancements to the algorithm are presented in Subsections 4.2 to 4.5.

4.1 A simple outer approximation algorithm

Since MILP solvers are more readily available (and often more reliable) than MINLP solvers, we decided to solve the SPARC by means of *Outer Approximation* (OA), which involves the solution of a series of progressively finer MILP relaxations of the original convex MINLP [6, 8]. The key idea is to approximate the convex constraints (12) with a collection of linear constraints of the form:

$$r_{ij} \;\leq\; f_i(\bar{p}) + f_i'(\bar{p})\,(p_{ij} \;-\; \bar{p}), \tag{14}$$

where the $\bar{p}$ values are selected from the domain $[0, P]$, and f_i' denotes the first derivative of f_i. The constraints (14) are called *Kelley cuts*, since they were first developed by Kelley [16] to solve convex NLPs.

A high-level description of a rudimentary OA algorithm for the SPARC is given in Algorithm 1. Here are some words of explanation:

- The algorithm requires two tolerance parameters. The parameter ϵ_1 is the minimum acceptable violation of a Kelley cut, expressed as a percentage of the right-hand side of constraint (12), and the parameter ϵ_2 is the maximum acceptable relative gap between the final values of the bounds U and L.

- For SPARC instances arising in practice, the N_i values can be very small (e.g., 10^{-12}). This means that the coefficient $f_i'(\bar{p})$ in the Kelley cut can be very large when $\bar{p}$ is close to zero, which can cause serious numerical difficulties. For this reason, instead of setting $\bar{p}$ to p_{ij}^*, we set it to $f^{-1}(r_{ij}^*)$, which is larger (see Figure 1).

- To construct our initial MILP relaxation, we include the objective function (9), the constraints (3), (4), (6)-(8), (11) and (13), and a collection of $|I|\,|J|$ Kelley cuts; namely, those Kelley cuts that are tight at the optimal solution to the continuous relaxation of (9)–(13).

Algorithm 1: Outer Approximation for SPARC

input : power P, bandwidths B_i, noise powers N_i,
data rate limits ℓ_j, tolerances ϵ_1, ϵ_2.

Set lower bound L to 0 and construct initial MILP;
repeat
  Solve the current MILP;
  if *the MILP is infeasible* **then**
    Output "The instance is infeasible." and quit;
  end
  Let (x^*, p^*, r^*) be the optimal solution to the MILP;
  Let U be the associated upper bound;
  Solve an NLP to find the best p for the given x^*;
  if *the NLP is feasible* **then**
    Let p' be the optimal NLP solution;
    Let L' be the associated profit;
  end
  if $L' > L$ **then**
    Set L to L' and save the incumbent solution (x^*, p');
  end
  for *all* $i \in I$ *and* $j \in J$ *such that* $x^*_{ij} = 1$ **do**
    if *the constraint (12) is violated by more than* ϵ_1 **then**
      Let $\bar{p} = f^{-1}(r^*_{ij})$;
      Generate the Kelley cut (14) for the given i, j and $\bar{p}$;
      Add the cut to the MILP;
    end
  end
until $L > 0$ *and* $(U - L)/L \leq \epsilon_2$;

output: A near-optimal solution (x^*, p') or an infeasibility warning.

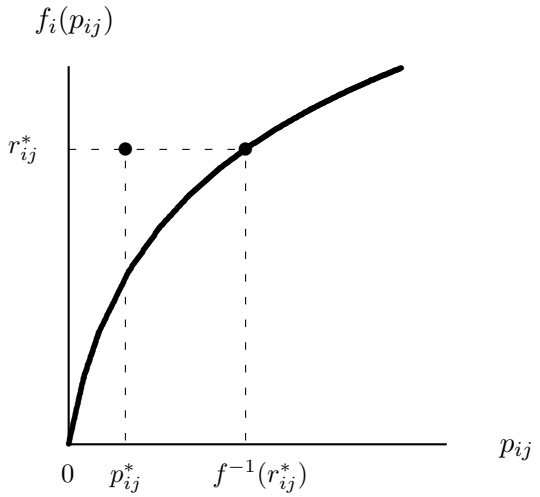

Figure 1: Choosing $\bar{p}$ in order to avoid numerical issues.

- The best SPARC solution found so far, if any, is called the "incumbent". After each MILP relaxation has been solved, we attempt to find a new incumbent by solving the NLP:

$$\max \left\{ \sum_{i \in I} f_i(p_i) : \sum_{i \in I} p_i \leq P, \sum_{i \in S_j} f_i(p_i) \geq \ell_j \ (j \in J), \ p \in \mathbb{R}_+^{|I|} \right\},$$

  where $S_j = \{i \in I : x^*_{ij} = 1\}$ is the set of subcarriers that were allocated to user j in the MILP solution. Since this NLP has only $|I|$ variables, it is usually solved very quickly.

Our preliminary experiments with this OA algorithm revealed that it struggled to solve even very small SPARC instances. Fortunately, we were able to improve the algorithm dramatically with four modifications. These are described in the following four subsections.

4.2 Perspective cuts

The main problem with the OA algorithm turned out to be that the MILPs had extremely weak continuous relaxations. To strengthen them, we used the following ideas of Frangione & Gentile [11].

Consider a convex MINLP in which the objective or constraints contain a term $f(y)$, where y is a vector of continuous variables and f is a convex function. Suppose that the convex MINLP also contains a binary variable x, with the property that, if x takes the value zero, then all of the components of y must also take the value zero. Then the continuous relaxation of the convex MINLP is strengthened if we replace the function $f(y)$ with the

perspective function $xf(y/x)$. The effect of this strengthening on an OA algorithm is as follows. Let z be a continuous variable representing the function $f(y)$, and let

$$z \geq f(\bar{y}) + \nabla f(\bar{y}) \cdot (y - \bar{y}),$$

be the associated Kelley cuts. Letting z represent the perspective function $xf(y/x)$ instead, the Kelley cuts change to:

$$z \geq \nabla f(\bar{y}) \cdot y + \Big(f(\bar{y}) - \nabla f(\bar{y}) \cdot \bar{y}\Big)x.$$

These cutting planes are called *perspective cuts*. Note that, when $x = 1$, they reduce to Kelley cuts, but when $x < 1$, they are stronger.

To apply this idea to the SPARC, observe that, for all $i \in I$ and $j \in J$, the continuous variable p_{ij} must be zero whenever x_{ij} is zero. Accordingly, we can replace the Kelley cuts (14) with the perspective cuts

$$r_{ij} \;\leq\; f_i'(\bar{p})\, p_{ij} \;+\; \Big(f_i(\bar{p}) - f_i'(\bar{p})\, \bar{p}\Big)\, x_{ij} \quad (\forall i \in I,\, j \in J,\, \bar{p} \in [0, P]). \qquad (15)$$

We found that using perspective cuts in place of Kelley cuts improved the running time of the MILP solver, and therefore of the whole OA algorithm, by two orders of magnitude (for given values of the tolerance parameters ϵ_1, ϵ_2).

We believe that this is the first time that perspective cuts have been applied to an optimisation problem from mobile wireless communications. For applications to problems in *wired* communications, see, e.g., Frangioni *et al.* [9, 10]. Applications in location, scheduling, network design, finance and power generation are surveyed in Günlük & Linderoth [13].

4.3 Pre-Emptive Cut Generation

In our experiments with the OA algorithm, we noticed the following phenomenon. The upper bound U would remain virtually unchanged for several iterations, then decrease, then remain virtually unchanged for several iterations, and so on. The cause of this turned out to be *symmetry*, or, more precisely, *near-symmetry*, among users. (See, e.g., Margot [20] for an introduction to symmetry issues in integer programming.)

Consider a fixed subcarrier $i \in I$, and suppose that $x^*_{ij} = 1$ in the optimal solution to the current MILP relaxation. If $r^*_{ij} > f_i(p^*_{ij})$, then a perspective cut is generated for the given i and j. In the next iteration, however, the MILP solver simply selects a user j' such that $\ell_{j'}$ is similar to ℓ_j, sets $x_{ij'} = 1$, and sets $p_{ij'}$ and $r_{ij'}$ to the values that p_{ij} and r_{ij} had before. If this happens for all $i \in I$, then the upper bound does not decrease even after adding a whole round of perspective cuts. In the worst case, when all of the ℓ_j values are similar, the upper bound will decrease only after $|J|$

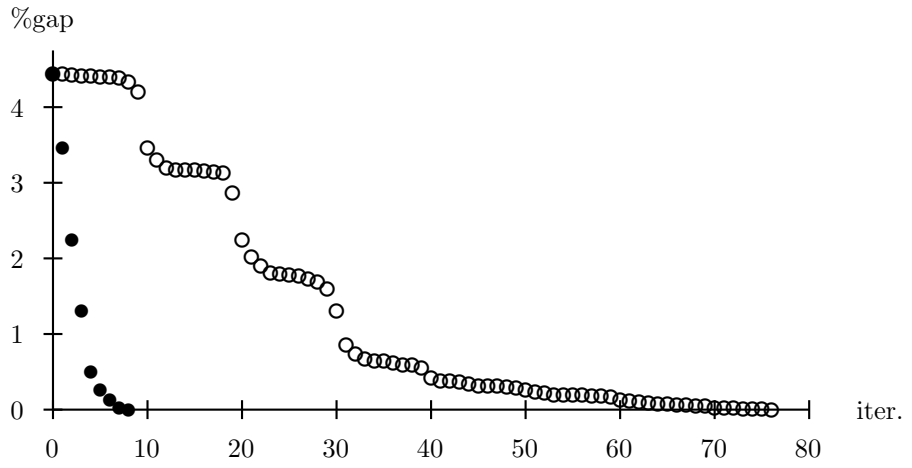

Figure 2: Typical evolution of percentage gap between upper bound and optimum, without pre-emptive cut generation (hollow circles) and with pre-emptive cut generation (filled circles).

MILPs have been solved, i.e., only after a perspective cut has been added for all pairs i and j.

We experimented with several ways to address this symmetry problem. In the end, the most effective approach was to generate more cuts in each major OA iteration. Specifically, in Algorithm 1, we replaced the line

"Generate the Kelley cut (14) for the given i, j and $\bar{p}$"

with the line

"For all $j \in J$, generate the perspective cut (15) for the given i, j and $\bar{p}$."

Note that the additional cuts are not violated in the current iteration, but are likely to be violated in future iterations. For this reason, we call this technique *pre-emptive cut generation* (PCG). We found that PCG reduced the number of OA iterations, and therefore the running time of the whole OA algorithm, by at least an order of magnitude (again, for given values of ϵ_1, ϵ_2).

Figure 2 demonstrates the benefits of PCG. It shows the evolution of the percentage gap between the upper bound and the optimal objective value, for a random instance with $|I| = 36$ and $|J| = 10$, both with and without PCG. Without PCG, over 70 iterations are needed to obtain an upper bound within 0.1% of optimal. With PCG, only 8 iterations are needed. A similar benefit is obtained with regard to running time. We remark that the benefit of PCG increases as the number of users increases.

4.4 Pre-processing

Thirdly, and perhaps most surprisingly, we discovered that many SPARC instances can be solved very quickly with some (relatively) straightforward procedures, which we refer to as *pre-processing*. Our pre-processing consists of three phases: an upper-bound computation, an infeasibility test, and a primal heuristic.

In the first phase, we relax the SPARC by dropping the rate constraints. The assignment of subcarriers to users then becomes irrelevant, and only the power allocation matters. Accordingly, we can compute an upper bound for the SPARC by solving the NLP (1). Given that the NLP is convex and separable, and has only $|I|$ variables, one can expect to solve it much more quickly than the SPARC itself. We let p^* denote the optimal solution of the NLP, and let $U = \sum_{i \in I} f_i(p_i^*)$ be the associated upper bound.

The second phase is a quick test for infeasibility. The idea is that, if $U < \sum_{j \in J} \ell_j$, then the SPARC instance must be infeasible, since there is no way to satisfy all of the rate constraints simultaneously. In that case, we can stop immediately.

The final phase is based on the following observation: since the $f_i(p_i^*)$ achieves the optimal transmission rate, if we can find an allocation of channels to users that meets all rate constraints, it must be optimal, and we can again stop immediately. We experimented with constructive heuristics for finding such a solution. In the end, however, it turned out to be best simply to feed the following 0-1 LP into an MILP solver:

$$\begin{array}{lll} \max & \sum_{i \in I} \sum_{j \in J} f_i(p_i^*) x_{ij} & \\ & \sum_{j \in J} x_{ij} \leq 1 & (\forall i \in I) \\ & \sum_{i \in I} f_i(p_i^*) x_{ij} \geq \ell_j & (\forall j \in J) \\ & x_{ij} \in \{0, 1\} & (\forall i \in I,\, j \in J). \end{array}$$

One can check that this 0-1 LP is feasible if and only if there exists a SPARC solution with profit equal to U. (Details on the MILP solver and parameter settings are given at the start of Section 5.)

In general, one can expect our pre-processing routines to be effective when the ℓ_j values are either very large (in which case the instance is quickly proven infeasible) or reasonably small (in which case the pre-processing algorithm can easily find an optimal solution). The results in Section 5 show that, in fact, the range of ℓ_j values for which pre-processing fails is very narrow.

4.5 Warm-starting

Our fourth and final improvement is concerned with *warm-starting* the OA algorithm. In our preliminary experiments with the algorithm, we noticed

that some of the r^*_{ij} values started out very high and then decreased very slowly from one iteration to the next. Investigation of the output revealed the following:

- The reason for the initial high r^*_{ij} values is that, in the optimal solution to the continuous relaxation of (9)–(13), all x variables take the value $1/|J|$, and all p variables take very small values (typically close to $P/(|I|\,|J|)$). This in turn is due to the very high slope of the functions $f_i(p_{ij})$ near zero. As a result, the initial family of perspective cuts is generated with excessively small values of $\bar{p}$.

- The reason for the slow decrease was caused by our "cautious" rule for selecting $\bar{p}$ when generating additional cuts (see Subsection 4.1). That is, it tends to generate cuts with rather large values of $\bar{p}$ in the early iterations of the OA algorithm.

In order to address this issue, we decided to use a different rule for selecting the initial set of perspective cuts. Specifically, for a given $i \in I$ and $j \in J$, we include three cuts, with $\bar{p}$ set to each of:

- P, the maximum value possible;

- p^*_i, where p^* is the vector obtained in the first pre-processing phase (see the last subsection);

- the harmonic mean of p^*_i and P, i.e., $\sqrt{p^*_i P}$.

We found that this change led to a roughly 40% additional reduction in both the number of OA iterations and the overall running time.

5 Computational Experiments

The enhanced Outer Approximation algorithm was coded in Julia v0.5 and run on a virtual machine cluster with 16 CPUs (ranging from Sandy Bridge to Haswell architectures) and 16GB of RAM, under Ubuntu 16.04.1 LTS. The program calls on MOSEK 7.1 (with default settings) to solve the NLP (1) in the first pre-processing phase, and on the mixed-integer solver from the CPLEX 12.6.3 Callable Library (again with default settings) to solve the MILP relaxations. We also used the mixed-integer solver to solve the 0-1 LP in the third pre-processing phase, but with the parameter `MIPemphasis` set to "emphasize feasibility" and a time limit of 5 seconds imposed. Finally, both tolerance parameters ϵ_1, ϵ_2 were set to 0.1%.

5.1 Test instances

For our batch of experiments, the number of subcarriers, $|I|$, was set to 72, the noise powers N_i were set to random numbers distributed uniformly in

$(0, 10^{-11})$, and the power limit P was set to 36W, i.e., 0.5W per subcarrier. These figures are typical of a small (typically indoor) base station. Following the IEEE 802.16 standard, the bandwidths B_i were all set to 1.25MHz. We considered four values for the number of users, $|J|$: 4, 6, 8 and 10.

Generating suitable user demands (i.e., ℓ_j values) turned out to be more difficult, for the following reason. Consider the initial upper bound U computed in the pre-processing phase (Subsection 4.4), along with the quantity

$$\frac{\sum_{j \in J} \ell_j}{U},$$

which we call the *demand ratio* (DR) of the given SPARC instance. If the DR exceeds 1, then the instance is immediately detected to be infeasible in phase 2 of the pre-processing procedure. On the other hand, if the DR is much smaller than 1, then an optimal solution to the instance is easily found in phase 3 of the pre-processing procedure. Thus, the DR is a critical parameter in determining the difficulty of an instance. Unfortunately, the DR cannot be known without solving the NLP (1).

This led us to use the following rather complicated procedure to generate the ℓ_j values. For a given $|I|$, $|J|$, B, N and P, we solve the NLP (1) to obtain U. Then, for all $j \in J$, we generate a random number z_j, which follows the unit lognormal distribution. (That is, $z_j = e^{t_j}$, where t_j is Normally distributed with zero mean and unit variance.) We then use the formula:

$$\ell_j = \frac{z_j * DR * U}{\sum_{k \in J} z_k}.$$

One can check that this procedure yields instances with the desired DR. We considered DR values from 0.90 to 0.99 in steps of 0.01. For each combination of $|J|$ and DR, we generated 500 random instances. This makes $4 \times 10 \times 500 = 20,000$ instances in total.

5.2 Experimental results

We start by presenting results obtained with pre-processing alone. We were surprised to find that all instances with DR below 0.96, and many instances with higher DR value, could be solved by pre-processing within the 5s time limit. Table 1 shows, for various combinations of $|J|$ and DR, the number of instances (out of 500) that could not be solved by pre-processing.

We see that, as expected, pre-processing grows less effective as the DR approaches 1 from below. Interestingly, it also tends to get less effective as $|J|$ increases. This is probably because, as $|J|$ increases, each user is allocated fewer subcarriers, which gives the pre-processing algorithm fewer opportunities to meet the demand of each user.

We remark that pre-processing was very fast for the majority of the instances. In about 80% of the cases, it took less than 0.1s. In about 97% of the cases, it took less than 1s. The time limit of 5s was rarely exceeded.

Table 1: Number of instances not solved by pre-processing

	Demand Ratio									
$\|J\|$	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
4	0	0	0	0	0	0	0	0	0	15
6	0	0	0	0	0	0	0	0	2	71
8	0	0	0	0	0	0	0	4	43	250
10	0	0	0	0	0	0	1	47	209	435

Table 2: Number of instances proved infeasible by OA algorithm

	Demand Ratio		
$\|J\|$	0.97	0.98	0.99
4	—	—	9
6	—	1	24
8	—	2	56
10	—	8	81

We call the instances that were unsolved by pre-processing *hard*. For each hard instance, we ran our exact OA algorithm until either (a) the gap between the upper bound U and the lower bound L, measured as a percentage of L, dropped below 0.1%, (b) the instance was proved to be infeasible, or (c) we exceeded a time limit of 120 seconds.

In Table 2, we report the number of hard instances proved to be *infeasible* by the OA algorithm. As expected, when DR and/or $|J|$ take higher values, the proportion of infeasible instances increases. We remark that the average time taken to prove infeasibility of these instances was about 1.35s.

Finally, in Table 3, we report the number of *feasible* hard instances solved within the time limit, and the mean time (in seconds) taken to solve them. We see that the OA algorithm solves many of these instances quickly, but runs into difficulty for large values of DR and/or $|J|$.

Table 3: Number of instances solved / Mean time taken by OA algorithm

	Demand Ratio				
$\|J\|$	0.95	0.96	0.97	0.98	0.99
4	—	—	—	—	4/35.65
6	—	—	—	1/30.93	40/36.77
8	—	—	3/64.37	33/67.51	116/53.24
10	—	1/31.25	22/58.66	80/66.57	54/67.78

The upshot of all this is that the majority of the instances could be solved in less than a second with the pre-processing algorithm, and most of the remaining instances could be solved (or proved infeasible) in about one minute by the OA algorithm. This means that our algorithms are potentially of practical use, especially in a so-called *slow fading* environment, where users switch base stations infrequently (as long as DR is not unusually close to 1).

5.3 Comparison with BONMIN

Finally, we compared our algorithm with the open-source MINLP solver BONMIN [2]. In our initial experiments, we tried feeding four different formulations to BONMIN: the original formulation (2)–(8), the modified formulation (9)–(13), and the formulations obtained from those by replacing the functions $f_i(p_{ij})$ with their perspective functions $x_{ij}f_i(p_{ij}/x_{ij})$. It turned out that BONMIN consistently solved the second formulation at least one order of magnitude faster than the first, and generated an error message (presumably due to division by zero) when attempting to solve either of the perspective-based formulations.

Surprisingly, even with the best formulation, BONMIN struggled to solve most of the 20,000 instances that we mentioned above. Thus, we created some simpler instances. Specifically, for four combinations of channels and users, we created instances with DR set to 20%, 50% and 99%. This makes 12 instances in total.

Table 4 presents details of the 12 instances, along with the running times in seconds for four algorithms: B-OA (Outer Approximation from BONMIN with MILP sub-solver set to CPLEX), B-BB (simple branch-and-bound from BONMIN), our pre-processing procedure, and our OA algorithm. A time limit of 12 hours was applied to all the algorithms. We mark instances where the time limit was exceed as "TL", instances where BONMIN wrongly concluded that a local optimum was a global optimum as as "LOC", and instances not solved by pre-processing as "NS".

Clearly, BONMIN struggles to solve the SPARC. We think that this is due mainly to (i) the ill-conditioning of the functions $f_i(p_{ij})$ at zero mentioned in Subsection 4.1, and (ii) the high degree of near-symmetry, as discussed in Subsection 4.3.

6 Conclusion

Joint subcarrier and power allocation problems arise frequently in mobile wireless communications. For one such problem, that we have called the SPARC, we have shown that it is possible to devise an effective exact algorithm based on intelligent pre-processing and a judicious use of known

Table 4: Running times of our algorithms compared with two algorithms of BONMIN (when using formulation (9)–(13))

		Algorithm			
$\|I\|/\|J\|$	DR	B-OA	B-BB	Pre-Processing	OA Exact
10/2	20%	22.56	387.51	0.01	0.06
	50%	115.75	352.10	0.01	0.14
	99%	253.52	5328.88	0.01	0.19
10/4	20%	42390.65	TL	0.01	0.07
	50%	11578.19	TL	0.01	0.31
	99%	17806.97	TL	NS	1.28
72/2	20%	LOC	TL	0.01	0.49
	50%	LOC	TL	0.01	0.50
	99%	TL	TL	0.01	0.70
72/4	20%	LOC	TL	0.01	1.45
	50%	LOC	TL	0.01	1.50
	99%	TL	TL	0.01	5.87

MINLP tools, including outer approximation, perspective cuts and symmetry-breaking.

There are several interesting topics for future research. First, we would like to prove that SPARC is $\mathcal{NP}$-hard in the strong sense. Second, it would be interesting to try using a "one-tree" approach, such as LP/NLP-based branch-and-bound, instead of OA (see [2, 21]). Third, it would be worth trying to adapt our exact algorithm to other resource allocation problems in mobile wireless systems, especially variants in which one wishes to maximise energy efficiency (see Subsection 2.2), or in which not all user demands can be met (see Subsection 3.2). Finally, one could consider how to allocate OFDMA resources dynamically in a fast-fading environment, in which users arrive and depart frequently in a stochastic manner.

References

[1] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke & A. Mahajan (2013) Mixed-integer nonlinear optimization. *Acta Numerica*, 22, 1–131.

[2] P. Bonami, L. Biegler, A. Conn, G. Cornuejols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot & W. A. (2008) An algorithmic framework

for convex mixed integer nonlinear programs. *Discr. Optim.*, 5, 186–204.

[3] S. Boyd & L. Vandenberghe (2004) *Convex Optimization*. Cambridge, UK: Cambridge University Press.

[4] T. Cover & J. Thomas (1991) *Elements of Information Theory*. New York: Wiley.

[5] C. D'Ambrosio & A. Lodi (2013) Mixed integer nonlinear programming tools: an updated practical overview. *Ann. Oper. Res.*, 204, 301–320.

[6] M. Duran & I. Grossmann (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.*, 36, 307–339.

[7] K. Fazel & S. Kaiser (2008) *Multi-Carrier and Spread Spectrum Systems (2nd edn.)*. Chichester: Wiley.

[8] R. Fletcher & S. Leyffer (1994) Solving mixed integer nonlinear programs by outer approximation. *Math. Program.*, 66, 327–349.

[9] A. Frangioni, L. Galli & M. Scutellà (2015) Delay-constrained shortest paths: approximation algorithms and second-order cone models. *J. Optim. Th. Appl.*, 164, 1051–1077.

[10] A. Frangioni, L. Galli & G. Stea (2015) Optimal joint path computation and rate allocation for real-time traffic. *Computer J.*, 58, 1416–1430.

[11] A. Frangioni & C. Gentile (2006) Perspective cuts for a class of convex 0-1 mixed integer programs. *Math. Program.*, 106, 225–236.

[12] X. Ge, J. Hu, C.X. Wang, C.H. Youn, J. Zhang & X. Yang (2012) Energy efficiency analysis of MISO-OFDM communication systems considering power and capacity constraints. *Mobile Netw. Appl.*, 17, 29–35.

[13] O. Günlük & J. Linderoth (2011) Perspective reformulation and applications. In J. Lee & S. Leyffer (eds.) *Mixed Integer Nonlinear Programming*, 61–89. Berlin: Springer.

[14] S. Haykin (1994) *Communication Systems*. New York: Wiley.

[15] C. Isheden, Z. Chong, E. Jorswieck & G. Fettweis (2012) Framework for link-level energy efficiency optimization with informed transmitter. *IEEE Trans. Wirel. Commun.*, 11, 2946–2957.

[16] J.E. Kelley, Jr. (1960) The cutting-plane method for solving convex programs. *SIAM J.*, 8, 703–712.

[17] K. Kim, Y. Han & S.L. Kim (2005) Joint subcarrier and power allocation in uplink OFDMA systems. *IEEE Commun. Lett.*, 9, 526–528.

[18] Y.F. Liu & Y.H. Dai (2014) On the complexity of joint subcarrier and power allocation for multi-user OFDMA systems. *IEEE Trans. Signal Process.*, 62, 583–596.

[19] Z.Q. Luo & S. Zhang (2008) Dynamic spectrum management: Complexity and duality. *IEEE J. Sel. Topics Signal Process.*, 2, 57–73.

[20] F. Margot (2010) Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi & L. Wolsey (eds.) *50 Years of Integer Programming 1958-2008*, 647–686. Berlin/Heidelberg: Springer.

[21] I. Quesada & I. Grossmann (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comput. Chem. Eng.*, 16, 937–947.

[22] M. Resende & P. Pardalos (eds.) (2007) *Handbook of Optimization in Telecommunications*. New York: Springer US.

[23] W. Rhee & J. Cioffi (2000) Increase in capacity of multiuser OFDM system using dynamic subchannel allocation. In *Proc. 2000 IEEE Veh. Technol. Conf.*, volume 2, 1085–1089.

[24] K. Seong, M. Mohseni & J. Cioffi (2006) Optimal resource allocation for OFDMA downlink systems. In *Proc. 2006 IEEE Inter. Symp. Inform. Th.*, 1394–1398.

[25] C. Shannon (1949) Communication in the presence of noise. *Proc. IRE*, 1, 10–21.

[26] Z. Shen, J.G. Andrews & B.L. Evans (2008) Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints. *IEEE Trans. Wirel. Commun.*, 4, 2726–2737.

[27] M. Tao, Y.C. Liang & F. Zhang (2008) Resource allocation for delay differentiated traffic in multiuser OFDM systems. *IEEE Trans. Wirel. Commun.*, 7, 2190–2201.

[28] M. Tawarmalani & N. Sahinidis (2005) A polyhedral branch-and-cut approach to global optimization. *Math. Program.*, 103, 225–249.

[29] T. Ting, S.F. Chien, X.S. Yang & S. Lee (2014) Analysis of quality-of-service aware orthogonal frequency division multiple access system considering energy efficiency. *IET Commun.*, 8, 1947–1954.

[30] F. Trespalacios & I. Grossmann (2014) Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86, 991–1012.

[31] C.Y. Wong & R.S. Cheng (1999) Multiuser OFDM with adaptive subcarrier, bit, and power allocation. *IEEE J. Sel. Areas Commun.*, 17, 1747–1758.

[32] X. Xiao, X. Tao & J. Lu (2013) QoS-aware energy-efficient radio resource scheduling in multi-user OFDMA systems. *IEEE Commun. Lett.*, 17, 75–78.

[33] C. Xiong, G. Li, S. Zhang, Y. Chen & S. Xu (2011) Energy-and spectral-efficiency tradeoff in downlink OFDMA networks. *IEEE Trans. Wirel. Commun.*, 10, 3874–3886.

[34] W. Yu & R. Lui (2006) Dual methods for nonconvex spectrum optimization of multicarrier systems. *IEEE Trans. Commun.*, 54, 1310–1322.

[35] C. Zarakovitis C & Q. Ni (2016) Maximising energy efficiency in multi-user multi-carrier broadband wireless systems: convex relaxation and global optimisation techniques. *IEEE Trans. Veh. Technol.*