A Heuristic for Maximising Energy Efficiency in an OFDMA System Subject to QoS Constraints

Adam N. Letchford[1], Qiang Ni[2], and Zhaoyu Zhong[1]

[1] Department of Management Science, Lancaster University,
Lancaster LA1 4YX, United Kingdom
{A.N.Letchford,z.zhong1}@lancaster.ac.uk
[2] School of Computing and Communications, Lancaster University,
Lancaster LA1 4WA, United Kingdom
q.ni@lancaster.ac.uk

Abstract. OFDMA is a popular coding scheme for mobile wireless multi-channel multi-user communication systems. In a previous paper, we used mixed-integer nonlinear programming to tackle the problem of maximising energy efficiency, subject to certain quality of service (QoS) constraints. In this paper, we present a heuristic for the same problem. Computational results show that the heuristic is at least two orders of magnitude faster than the exact algorithm, yet yields solutions of comparable quality.

Keywords: OFDMA systems, energy efficiency, heuristics

1 Introduction

In many mobile wireless communications systems, mobile devices communicate with one another via transceivers called *base stations*. Many base stations follow the so-called *Orthogonal Frequency-Division Multiple Access* (OFDMA) scheme to code and transmit messages (see, e.g., [3]). In OFDMA, we have a set of communication channels, called *subcarriers*, and a set of *users* (i.e., mobile devices that are currently allocated to the given base station). Each subcarrier can be assigned to at most one user, but a user may be assigned to more than one subcarrier. The data rate achieved by any given subcarrier is a nonlinear function of the power allocated to it.

Several different optimisation problems have been defined in connection with OFDMA systems (e.g., [6–11,15,17–21]). Unfortunately, it turns out that most of these problems are $\mathcal{NP}$-hard [5,10,11]. Thus, most authors resort to heuristics. In our recent papers [8,9], however, we presented exact solution algorithms based on mixed-integer nonlinear programming (MINLP). The problem considered in [9] is to maximise the total data rate of the system subject to certain quality of service (QoS) constraints called *user rate* constraints. The one considered in [8] is similar, except that the objective is to maximise the energy efficiency (defined as the total data rate divided by the total power used).

The algorithm in [9] is capable of solving many realistic problem instances to proven optimality (or near-optimality) within a couple of seconds. The algorithm in [8], however, is a lot slower, taking several minutes in some cases. This makes it of little use in a highly dynamic environment, when users may arrive and depart frequently at random. Thus, we were motivated to devise a fast heuristic for the problem described in [8]. That heuristic is the topic of the present paper.

Our heuristic is based on a combination of fractional programming, 0-1 linear programming and binary search. It turns out to be remarkably effective, being able to solve realistic instances to within 1% of optimality within a few seconds.

The paper has a simple structure. The problem is described in Section 2, the heuristic is presented in Section 3, the computational results are given in Section 4, and concluding remarks are made in Section 5.

To make the paper self-contained, we recall the following result from [13] (see also [1, 14]). Consider a *fractional program* of the form:

$$\max \Big\{ f(y)/g(y) : \, y \in C \Big\},$$

where $C \subseteq \mathbb{R}^n$ is convex, $f(y)$ is non-negative and concave over the domain C, and $g(y)$ is positive and convex over C. This problem can be reformulated as

$$\max \Big\{ t f(y'/t) : \, t g(y'/t) \leq 1, \; y' \in tC, \; t > 0 \Big\},$$

where t is a new continuous variable representing $1/g(y)$, and y' is a new vector of variables representing $y/g(y)$. The reformulated problem has a concave objective function and a convex feasible region.

2 The Problem

The problem under consideration is as follows. We have a set I of subcarriers and a set J of users, a (positive real) *system power* σ (measured in watts) and a total power limit P (also in watts). For each $i \in I$, we are given a *bandwidth* B_i (in megahertz), and a *noise power* N_i (in watts). Finally, for each $j \in J$, we are given a *demand* d_j (in megabits per second). The classical *Shannon-Hartley theorem* [16] implies that, if we allocate p units of power to subcarrier i, the data rate of that subcarrier (again in Mb/s) cannot exceed

$$f_i(p) = B_i \log_2 (1 \, + \, p/N_i) \, .$$

The task is to simultaneously allocate the power to the subcarriers, and the subcarriers to the users, so that energy efficiency is maximised and the demand of each user is satisfied.

In [8], this problem was called the *fractional subcarrier and power allocation problem with rate constraints* or F-SPARC. It was formulated as a *mixed 0-1 nonlinear program*, as follows. For all $i \in I$ and $j \in J$, let the binary variable x_{ij} indicate whether user j is assigned to subcarrier i, let the non-negative variable

p_{ij} represent the amount of power supplied to subcarrier i to serve user j, and let r_{ij} denote the associated data rate. The formulation is then:

$$\max \quad \frac{\sum_{i\in I}\sum_{j\in J} r_{ij}}{\sigma+\sum_{i\in I}\sum_{j\in J} p_{ij}} \tag{1}$$

$$\sum_{i\in I}\sum_{j\in J} p_{ij} \le P-\sigma \tag{2}$$

$$\sum_{j\in J} x_{ij} \le 1 \qquad (\forall i\in I) \tag{3}$$

$$\sum_{i\in I} r_{ij} \ge d_j \qquad (\forall j\in J) \tag{4}$$

$$r_{ij} \le f_i(p_{ij}) \qquad (\forall j\in J) \tag{5}$$

$$p_{ij} \le (P-\sigma)x_{ij} \qquad (\forall i\in I,\, j\in J) \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad (\forall i\in I,\, j\in J)$$

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

The objective function (1) represents the total data rate divided by the total power (including the system power). The constraint (2) enforces the limit on the total power. Constraints (3) ensure that each subcarrier is assigned to at most one user. Constraints (4) ensure that user demands are met. Constraints (5) ensure that the data rate for each subcarrier does not exceed the theoretical limit. Constraints (6) ensure that p_{ij} cannot be positive unless x_{ij} is one. The remaining constraints are just binary and non-negativity conditions.

The objective function (1) and the constraints (5) are both nonlinear, but they are easily shown to be concave and convex, respectively. The exact algorithm in [8] starts by applying the transformation mentioned at the end of the introduction, to make the objective function separable. After that, it uses a well-known generic exact method for convex MINLP, called *LP/NLP-based branch-and-bound* [12], enhanced with some specialised cutting planes called *bi-perspective* cuts. As mentioned in the introduction, however, this exact method can be too slow on some instances of practical interest.

3 The Heuristic

We now present our heuristic for the F-SPARC. We will show in the next section that it is capable of solving many F-SPARC instances to proven near-optimality very quickly.

3.1 The Basic Idea

Let $D = \sum_{j\in J} d_j$ be the sum of the user demands. We start by solving the following NLP:

$$\max\left\{\sum_{i\in I} f_i(p_i) : \sum_{i\in I} p_i \le P-\sigma,\, p\in\mathbb{R}_+^{|I|}\right\}. \tag{NLP1}$$

This gives the maximum possible data rate of the system, which we denote by M. If $D > M$, the F-SPARC instance is infeasible, and we terminate immediately.

We remark that `NLP1` can be solved extremely quickly in practice, since its objective function is both concave and separable. (In fact, it can be solved by the well-known *water-filling* approach; see, e.g., [2, 4].)

Now consider the following fractional program:

$$\begin{array}{ll} \max & \sum_{i\in I} f_i(p_i)/(\sigma + \sum_{i\in I} p_i) \\ \text{s.t.} & \sum_{i\in I} f_i(p_i) \geq D \\ & \sum_{i\in I} p_i \leq P - \sigma \\ & p_i \geq 0 \qquad (i \in I). \end{array}$$

This is a relaxation of the F-SPARC instance, since it ignores the allocation of subcarriers to users, and aggregates the user demand constraints. Using the transformation mentioned in the introduction, it can be converted into the following equivalent convex NLP:

$$\begin{array}{ll} \max & \sum_{i\in I} t f_i(\tilde{p}_i/t) \\ \text{s.t.} & \sigma t + \sum_{i\in I} \tilde{p}_i = 1 \\ & 1/P \leq t \leq 1/\sigma \\ & \sum_{i\in I} t f_i(\tilde{p}_i/t) \geq D\, t \\ & \tilde{p}_i \geq 0 \qquad (i \in I). \end{array} \tag{NLP2}$$

The solution of `NLP2` yields an upper bound on the efficiency of the optimal F-SPARC solution, which we denote by U.

Now, if we can find an F-SPARC solution whose efficiency is equal to U, it must be optimal. In an attempt to find such a solution, one can take the optimal solution of `NLP2`, say $(\tilde{p}^*, t^*)$, construct the associated data rate $r^*_i = f_i(\tilde{p}^*_i/t^*)/t^*$ for all $i \in I$, and then solve the following 0-1 linear program by branch-and-bound:

$$\begin{array}{ll} \max & \sum_{i\in I}\sum_{j\in J} x_{ij} \\ \text{s.t.} & \sum_{j\in J} x_{ij} \leq 1 \quad (i \in I) \\ & \sum_{i\in I} r^*_i\, x_{ij} \geq d_j \ (j \in J) \\ & x_{ij} \in \{0,1\} \end{array} \tag{01LP}$$

Note that `01LP`, having $|I|\,|J|$ variables, is of non-trivial size. On the other hand, all feasible solutions (if any exist) represent optimal F-SPARC solutions. Thus, if any feasible solution is found during the branch-and-bound process, we can terminate branch-and-bound immediately.

3.2 Improving with Binary Search

Unfortunately, in practice, `01LP` frequently turns out to be infeasible. This is because the sum of the r^*_i is frequently equal to D, which in turn means that a feasible solution of `01LP` would have to satisfy all of the linear constraints at perfect equality. Given that the r^*_i and d_j are typically fractional, such a solution is very unlikely to exist. (In fact, even if such a solution did exist, it could well be lost due to rounding errors during the branch-and-bound process.)

These considerations led us to use a more complex approach. We define a modified version of `NLP2`, in which D is replaced by $(1+\epsilon)\,D$, for some small

$\epsilon > 0$. We will call this modified version "NLP2(ϵ)". Solving NLP2(ϵ) in place of NLP2 usually leads to a small deterioration in efficiency, but it also tends to lead to slightly larger r_i^* values, which increases the chance that 01LP will find a feasible solution.

We found that, in fact, even better results can be obtained by performing a binary search to find the best value of ϵ. The resulting heuristic is described in Algorithm 1. When Algorithm 1 terminates, if L and U are sufficiently close (say, within 1%), then we have solved the instance (to the desired tolerance).

Algorithm 1: Binary search heuristic for F-SPARC

```
input: power P, bandwidths B_i, noise powers N_i,
       demands d_j, system power σ, tolerance δ > 0.
Compute total user demand D = Σ_{j∈J} d_j;
Solve NLP1 to compute the maximum possible data rate M;
Output D and M;
if D > M then
  | Print "The instance is infeasible." and quit;
end
Solve NLP2 to compute upper bound U on optimal efficiency;
Output U;
Set L := 0, ε_ℓ := 0 and ε_u := (M/D) − 1;
repeat
  | Set ε := (ε_ℓ + ε_u)/2;
  | Solve NLP2(ε). Let (p̃*, r̃*, t*) be the solution and L' its efficiency;
  | Solve 01LP with r* set to r̃*/t*;
  | if 01LP is infeasible then
  |   | Set ε_ℓ := ε;
  | else
  |   | Let x* be the solution to 01LP;
  |   | if L' > L then
  |   |   | Set L := L', p̄ := p̃*/t* and x̄ := x*;
  |   | end
  |   | Set ε_u := ε;
  | end
until ε_u − ε_ℓ ≤ δ or L ≥ U/(1 + δ);
if L > 0 then
  | Output feasible solution (x̄, p̄);
else
  | Output "No feasible solution was found.";
end
```

3.3 Improving by Reallocating Power

Algorithm 1 can be further enhanced as follows. Each time we find a feasible solution x^* to 01LP, we attempt to improve the efficiency of the associated F-

SPARC solution by solving the following fractional program:

$$\begin{array}{ll} \max & \sum_{i\in I} f_i(p_i)/(\sigma + \sum_{i\in I} p_i) \\ \text{s.t.} & \sum_{i\in I} p_i \le P - \sigma \\ & \sum_{i\in I: x^*_{ij}=1} f_i(p_i) \ge d_j \quad (j \in J) \\ & p_i \ge 0. \end{array}$$

This is equivalent to a convex NLP that is similar to NLP2, except that we replace the single constraint $\sum_{i\in I} \tilde{r}_i \ge D\, t$ with the "disaggregated" constraints

$$\sum_{i\in I: x^*_{ij}=1} \tilde{r}_i \ge d_j\, t \qquad (j \in J).$$

We call this modified NLP "NLP2dis". We found that this enhancement leads to a significant improvement in practice. Intuitively, it "repairs" much of the "damage" to the efficiency that was incurred by increasing the demand by a factor of $1 + \epsilon$.

4 Computational Experiments

We now report on some computational experiments that we conducted. The heuristic 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 NLPs, and on the mixed-integer solver from the CPLEX Callable Library (v. 12.6.3) to solve the 0–1 LPs. In CPLEX, default setting were used, except that the parameter "MIPemphasis" was set to "emphasize feasibility, and a time limit of 1 second was imposed for each branch-and-bound run. We also imposed a total time limit of 5 seconds for each F-SPARC instance.

4.1 Test Instances

To construct our test instances, we used the procedure described in [9], which is designed to produce instances typical of a small (indoor) base station following the IEEE 802.16 standard. These instances have $|I| = 72$, $|J| \in \{4, 6, 8\}$ and P set to 36 watts. The noise powers N_i are random numbers distributed uniformly in $(0, 10^{-11})$, and the bandwidths B_i are all set to 1.25MHz.

The user demands d_j are initially generated according to a unit lognormal distribution, and are then scaled to create instances of varying difficulty. Recall that, for a given instance, D denotes the total demand and M denotes the maximum possible data rate of the system. The quantity D/M is called the *demand ratio* (DR) of the instance. The user demands are scaled so that the DR takes values in $\{0.75, 0.8, 0.85, 0.9, 0.95, 0.98\}$. As the DR approaches 1 from below, the instances tend to get harder.

For each combination of $|J|$ and DR, we generated 500 random instances. This makes $3 \times 6 \times 500 = 9,000$ instances in total. For each instance, we first ran the exact algorithm in [8], with a tolerance of 0.01%, to compute tight upper bounds on the optimal efficiency. Although this was very time-consuming, it was necessary in order to assess the quality of the solutions found by our heuristic. We remark that some of the instances with high DR were proven to be infeasible by the exact algorithm.

4.2 Results

Table 1 shows, for various combinations of $|J|$ and DR, the number of instances (out of 500) for which the heuristic failed to find a feasible solution within the 5 second time limit. We see that the heuristic always finds a solution when the DR is less than 0.9, but can fail to find one when the DR is close to 1, especially when the number of users is high. This is however not surprising, since a high DR leads to fewer options when solving problem `01LP`, and an increase in $|J|$ increases the number of user demands that the heuristic needs to satisfy. Also, as mentioned above, some of the instances are actually infeasible.

	Demand Ratio					
$\|J\|$	0.75	0.80	0.85	0.90	0.95	0.98
4	0	0	0	0	5	9
6	0	0	0	7	22	29
8	0	0	0	10	59	82

Table 1. Number of instances where heuristic failed to find a feasible solution

Table 2 shows, for the same combinations of $|J|$ and DR, the average gap between the efficiency of the solution found by the heuristic, and our upper bound on the optimal efficiency. The average is taken over the instances for which the heuristic found a feasible solution. We see that the heuristic consistently finds an optimal solution when the DR is less than 0.85. Moreover, even for higher DR values, the solutions found by the heuristic are of excellent quality, with average gaps of well under 1%. (Closer inspection of the date revealed that the gap exceeded 1% only for some instances with DR equal to 0.9 or higher.)

	Demand Ratio					
$\|J\|$	0.75	0.80	0.85	0.90	0.95	0.98
4	0.00	0.00	0.30	0.24	0.33	0.38
6	0.00	0.00	0.31	0.32	0.48	0.54
8	0.00	0.00	0.33	0.41	0.67	0.90

Table 2. Average percentage gap between lower and upper bounds.

Finally, Table 3 shows the average time taken by the heuristic, again averaged over the instances for which the heuristic found a feasible solution. We see that, when the DR is less than 0.85, the heuristic finds the optimal solution within a fraction of a second. Moreover, even for higher DR values, the heuristic rarely needed the full 5 seconds allocated to it to find a solution of good quality. (Closer inspection of the data revealed that, in the majority of the cases in which the time limit was met, it was because the heuristic had not found a feasible solution by that time.)

	Demand Ratio					
$\lvert J \rvert$	0.75	0.80	0.85	0.90	0.95	0.98
4	0.07	0.06	0.11	0.13	0.28	0.38
6	0.07	0.06	0.36	0.57	1.10	1.40
8	0.07	0.09	0.58	0.90	2.00	2.83

Table 3. Average time (in seconds) taken by the heuristic when it succeeded.

All things considered, the heuristic performs very well, both in terms of solution quality and running time. Although we have not given detailed running times for the exact algorithm, we can report that the heuristic is typically faster by at least two orders of magnitude.

5 Concluding Remarks

Due to environmental considerations, it is becoming more and more common to take energy efficiency into account when designing and operating mobile wireless communications systems. We have presented a heuristic for one specific optimisation problem arising in this context, concerned with maximising energy efficiency in an OFDMA system. The computational results are very promising, with optimal or near-optimal solutions being found for the majority of instances in less than a second.

We believe that our heuristic is suitable for real-life application in a dynamic environment, as long as users arrive and depart only every few seconds. However, there is one caveat: our heuristic involves the solution of nonlinear programs (NLPs) and 0–1 linear programs (0-1 LPs), which in itself consumes energy. We believe that the NLP subproblems could be solved more quickly and efficiently using a specialised method (such as water-filling). As for the 0-1 LP subproblems, note that they are actually only *feasibility* problems, rather than optimisation problems *per se*. It may well be possible to solve them efficiently using a simple local-search heuristic, rather than invoking the "heavy machinery" of an exact 0-1 LP solver. This may be the topic of a future paper.

References

1. A. Charnes & W.W. Cooper (1962) Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4), 181–186.
2. T.M. Cover & J.A. Thomas (1991) *Elements of Information Theory*. New York: Wiley.
3. K. Fazel & S. Kaiser (2008) *Multi-Carrier and Spread Spectrum Systems*. Chichester: Wiley-Blackwell, 2nd edition.
4. S. Haykin (1994) *Communication Systems*. New York: Wiley.
5. P.H. Huang, Y. Gai, B. Krishnamachari & A. Sridharan (2010) Subcarrier allocation in multiuser OFDM systems: complexity and approximability. In *6th International Conference on Wireless Communications and Networking*. IEEE.
6. K. Kim, Y. Han & S.L. Kim (2005) Joint subcarrier and power allocation in uplink OFDMA systems. *IEEE Communications Letters*, 9(6), 526–528.
7. X. Lei & Z. Liang (2015) Joint time-frequency-power resource allocation algorithm for OFDMA systems. In *5th International Conference on Electronics Information and Emergency Communication*, 266–271. IEEE.
8. A.N. Letchford, Q. Ni & Z. Zhong (2017) Bi-perspective functions for mixed-integer fractional programs with indicator variables. Technical report, Department of Management Science, Lancaster University, UK.
9. A.N. Letchford, Q. Ni & Z. Zhong (2017) An exact algorithm for a resource allocation problem in mobile wireless communications. *Computational Optimization and Applications*, 68(2), 193–208.
10. Y.F. Liu & Y.H. Dai (2014) On the complexity of joint subcarrier and power allocation for multi-uuser OFDMA systems. *IEEE Transactions on Signal Processing*, 62(3), 583–596.
11. Z.Q. Luo & S. Zhang (2008) Dynamic spectrum management: complexity and duality. *IEEE Journal of Selected Topics in Signal Processing*, 2(1), 57–73.
12. I. Quesada & I.E. Grossmann (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16(10), 937–947.
13. S. Schaible (1974) Parameter-free convex equivalent and dual programs of fractional programming problems. *Zeitschrift für Operations Research*, 18(5), 187–196.
14. S. Schaible (1983) Fractional programming. *Zeitschrift für Operations Research*, 27(1), 39–54.
15. K. Seong, M. Mohseni & J. Cioffi (2006) Optimal resource allocation for OFDMA downlink systems. In *2006 IEEE International Symposium on Information Theory*, 1394–1398. IEEE.
16. C.E. Shannon (1949) Communication in the presence of noise. *Proceedings of the IRE*, 37(1), 10–21.
17. T.O. 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 Communications*, 8(11), 1947–1954.
18. X. Xiao, X. Tao & J. Lu (2013) QoS-aware energy-efficient radio resource scheduling in multi-user OFDMA systems. *IEEE Communications Letters*, 17(1), 75–78.
19. C. Xiong, G.Y. Li, S. Zhang, Y. Chen & S. Xu (2011) Energy- and spectral-efficiency tradeoff in downlink OFDMA networks. *IEEE Transactions on Wireless Communications*, 10(11), 3874–3886.
20. W. Yu & R. Lui (2006) Dual methods for nonconvex spectrum optimization of multicarrier systems. *IEEE Transactions on Communications*, 54(7), 1310–1322.

21. C.C. Zarakovitis & Q. Ni (2016) Maximizing energy efficiency in multiuser multicarrier broadband wireless systems: convex relaxation and global optimization techniques. *IEEE Transactions on Vehicular Technology*, 65(7), 5275–5286.