Optimal Resource Allocation In Base Stations for Mobile Wireless Communications

Zhaoyu Zhong

Management Science

Lancaster University Management School

A thesis submitted for the degree of

Doctor of Philosophy

May 2018

This thesis is dedicated to Jessica.

Thanks for your company

and

best wishes to you.

Acknowledgements

I would like to thank both my supervisors: Professor Adam Letchford and Qiang Ni, without whose help I could never have achieved this.

Professor Adam Letchford has been more than helpful throughout my whole PhD life. He helped me in my applications for admission and fundings, he also coached me programming and researching skills. I am so grateful to be one of his students, and words are powerless to express my gratitude.

Professor Qiang Ni brought us an exciting and inspiring topic to work on, based on which the four papers in this thesis were produced.

Also, thanks to all my colleagues and friends who made my PhD life not boring, but wonderful.

Papers Arising from the Research

We have included four papers produced during our research in this thesis as listed below. Those papers are presented as Chapters 2-5.

1. A.N. Letchford, Q. Ni & Z. Zhong (2017) An exact algorithm for a resource allocation problem in mobile wireless communications. *Cfomputational Optimization and Applications*, 68(2), 193-208.
2. A.N. Letchford, Q. Ni & Z. Zhong (2017) Bi-perspective functions for mixed-integer fractional programs with indicator variables. Revision submitted to *Mathematical Programming*
3. A.N. Letchford, Q. Ni & Z. Zhong (2018) A heuristic for maximising energy efficiency in an OFDMA system subject to QoS constraints. In J. Lee, G. Rinaldi & A.R. Mahjoub (eds.) *Combinatorial Optimization*, 303-312. Cham: Springer International Publishing.
4. A.N. Letchford, Q. Ni & Z. Zhong (2018) A Heuristic for Dynamic Resource Allocation in Overloaded OFDMA Systems. Submitted to *Journal of Heuristics*

Prof. Qiang Ni contributed the topic of optimisation in OFDMA systems and two classical cases where overall system transmission rate and energy efficiency are maximised respectively. Prof. Adam Letchford made a substantial contribution to all four papers by providing various ideas and mathematical proofs, especially the lemmas, theorems and proofs in the second paper. Inspired by Prof. Letchford, I programmed algorithms and performed experiments to assess all ideas and gave suggestions on modifications of algorithms.

Chapters 2-5 are joint-work by the three authors listed above. Chapters 1 and 6 are prepared by myself, and proofread by Prof. Letchford.

Abstract

Optimal Resource Allocation In Base Stations for Mobile Wireless Communications

Zhaoyu Zhong, B.Sc., M.Sc.

PhD thesis, May 2018

Department of Management Science

Lancaster University Management School

Telecommunications provides a rich source of interesting and often challenging optimisation problems. This thesis is concerned with a series of mixed-integer non-linear optimisation problems that arise in *mobile wireless* communications systems.

The problems under consideration arise when mobile base stations have an *Orthogonal Frequency-Division Multiple Access* (OFDMA) architecture, where there are subcarriers for data transmission and users with various transmission demands. In such systems, we simultaneously allocate subcarriers to users and power to subcarriers, subject to various constraints including certain quality of service (QoS) constraints called *rate constraints*. These problems can be modelled as *Mixed Integer Non-linear Programmes* (MINLP).

When we began the dissertation, we had the following main aims:

- To design an exact algorithm for the *subcarrier and power allocation problem with rate constraints* (SPARC), the objective of which is to maximise total data transmission rate of the entire system.
- To design an exact algorithm for the *fractional subcarrier and power allocation problem with rate constraints* (F-SPARC) problem in order to maximise system efficiency, i.e.: total data transmission rate divided by total power supplied to the system.

- To design a heuristic algorithm for the F-SPARC problem.
- To design a heuristic algorithm for the SPARC problem in dynamic settings, where user demand changes very frequently.

Along the way, however, we discovered a new approach to a broad family of problems, which includes the F-SPARC as a special case. These problems are called mixed-integer fractional programs with indicator variables, and they are dealt with in Chapter 3.

Contents

Chapter 1

Introduction

Mobile wireless devices include cell phones, smartphones and tablets. Since the 1980s, when the first generation of mobile wireless networks (1G) was established, the use of such devices has become far more widespread, to the point that they are now an indispensable part of a developed society.

1G networks were unstable, and only voice transmission was possible. 2G networks brought additional activities such as SMS and browsing text-only webpages. However, 2G networks only allowed data transmission at a peak rate of around 0.5Mbps. These networks pale in comparison to the peak rate of the current 4G networks which can exceed 300Mbps. Thanks to the rapid evolution in infrastructure, mobile devices are now capable of handling multi-media activities such as live streaming. Another recent development is the Internet of Things (IoT), in which everyday devices such as televisions, central heating systems or refrigerators, are connected to the Internet.

As the demand for higher transmission rates continues to climb, access to useful frequency bands in the electromagnetic spectrum (a.k.a. bandwidth) is becoming scarce and expensive. In this context, mathematical optimisation is of critical importance for finding improvements in both efficiency and effectiveness of bandwidth use. This is true for long-term (so-called "strategic"), medium-term ("tactical") and short-term ("operational") planning (see [64]).

In this thesis, we consider a family of operational problems which arise in mobile wireless communications. These problems are concerned with the allocation of resources (typically power and bandwidth) in order to handle incoming user requests. In the following sections of this chapter, we introduce some essential background and terminology for the two key subjects involved in the thesis: mobile wireless communications and optimisation. Concluding the chapter, an overview of the following chapters is given.

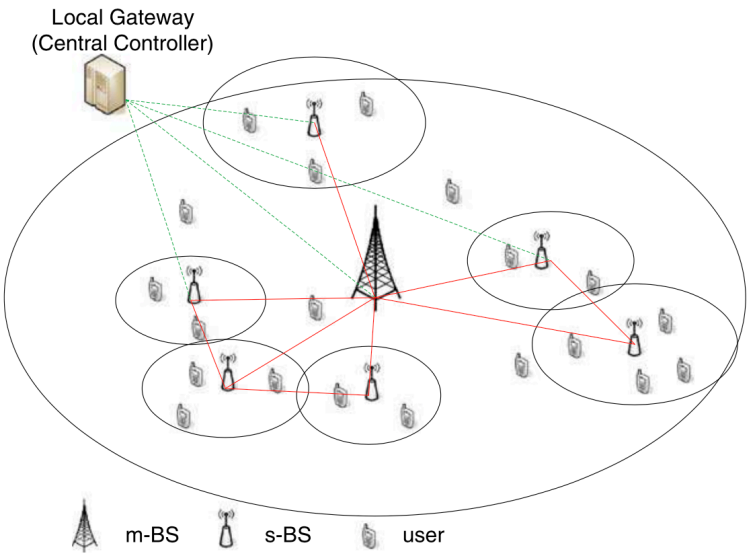


Figure 1.1: Cellular network model.

1.1 Mobile Wireless Communication

In mobile wireless networks, mobile devices (or users) transmit data over carrier waves and are connected to each other by base stations. A base station is a device (or building) with antennae to send and receive calls, texts and multi-media data to and from all devices within its coverage. Mobile phones transmit data to the base station which offers the strongest signal.

The coverage which base stations offer is dependent on many factors such as power supply and graphical characteristics. For best coverage, each base station is assigned a specific coverage area, called a *cell*. A mobile wireless network can be very complicated due to base stations of various sizes and controllers manipulating data transmission between regions. We show a cellular network model in Figure 1.1 with macro base stations (m-BS) and small base stations (s-BS). The cells are represented by small circles.

Cells of different base stations may slightly overlap in order to maximise integrated coverage. However, a mobile phone can only be connected to one base station at a time. Users may experience situations where calls are interrupted while traveling from one region to another. It's likely that the user is switching from one base station to another, and the latter base station needs time to allocate resources to the new user. Therefore, improving the allocation of resources such as power and carrier waves in mobile telecommunications has been of great concern during the past few decades.

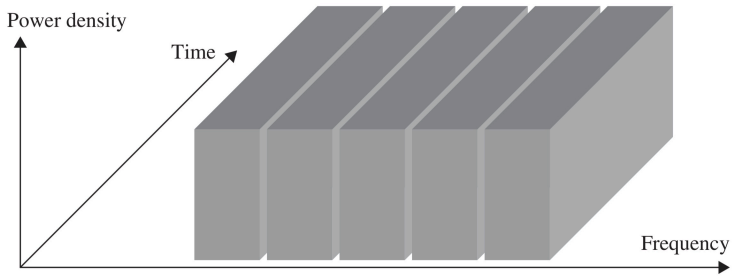


Figure 1.2: Principle of FDMA (with 5 sub-channels)

1.1.1 Evolution of Multiple Access Schemes

Multiplexing is the technology by which voice or data from multiple users can be transmitted on a shared frequency spectrum. Multiple access schemes are the interfaces applied in mobile networks based on a multiplexing technique. There have been a number of multiple access schemes applied during the history of mobile communications.

1G mobile networks used *Frequency-Division Multiple Access* (FDMA), which is based on *Frequency-Division Multiplexing* (FDM). In an FDMA system, data from each user are transmitted via one or several allocated individual sub-channels (see Figure 1.2). Each sub-channel is on a dedicated frequency, and can be only used by one user. In order to prevent interference between sub-channels, there are gaps between each pair of sub-channels. These are called guard bands. Despite FDMA having advantages such as simple implementation, its disadvantages far outweigh them. Guard bands between sub-channels, used to avoid interference, result in a waste of spectrum. Furthermore, the quality of transmission in 1G analogue networks was not guaranteed when users were physically close to each other. FMDA soon became redundant in mobile wireless communications.

The second generation of mobile wireless communications transited from analogue to digital technology. *Time-Division Multiple Access* (TDMA), a scheme based on *Time-Division Multiplexing* (TDM), was applied in 2G networks enabling simple data transmission services. In TDMA systems, data of all users are transmitted over the same frequency but in individual short time slots. In each time slot, the data of only one user can be transmitted (see Figure 1.3. TDMA allowed transmission of data from multiple users on the same frequency, but not simultaneously.

Code-Division Multiple Access (CDMA), based on *Code-Division Multiplexing* (CDM), had been implemented in some 2G networks before it came into the main-

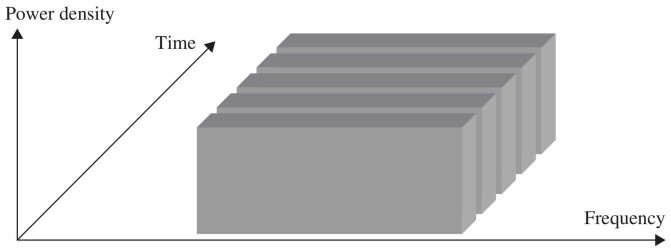


Figure 1.3: Principle of TDMA (with 5 time-slots)

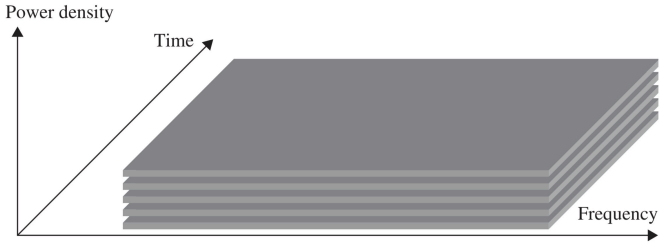


Figure 1.4: Principle of CDMA (with 5 spreading codes)

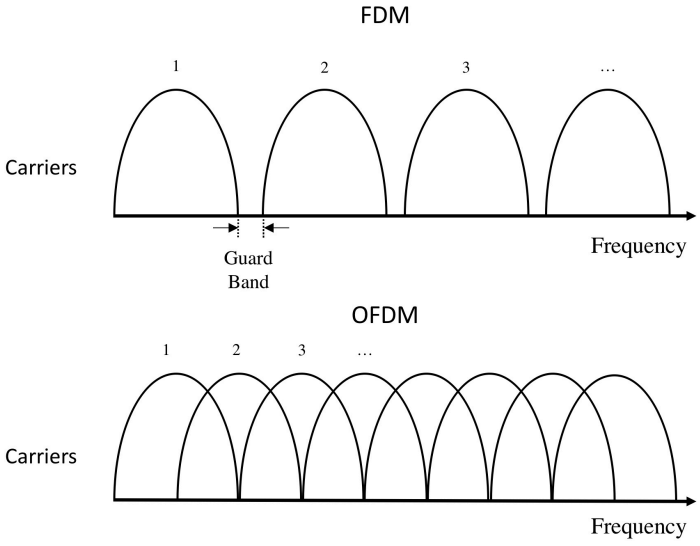


Figure 1.5: Bandwidth utilisation in FDM and OFDM multiplexing

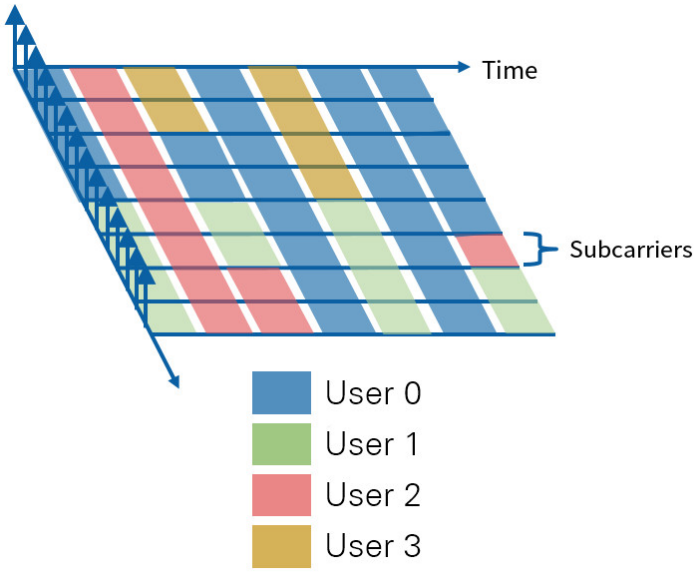


Figure 1.6: OFDMA subcarriers allocation over time

stream during the 3G era. CDMA enables data of more than one user to be transmitted on the same frequency, at the same time, by allocating a random spreading code to each user (see Figure 1.4). In a CDMA network, data from all users are transmitted at the same rate. This is not an optimal solution due to users having varying demands.

As the demand for higher data transmission rate increased, the *Orthogonal Frequency-Division Multiplexing* (OFDM) technique was proposed. OFDM is based on FDM. However, a frequency is divided into many orthogonal carriers to carry an information stream. All carriers are orthogonal to one another to avoid interference, so on a fixed bandwidth there can be more carriers than FDM carriers for data transmission (see Figure 1.5). Meanwhile, the orthogonal distribution of carriers eliminated guard bands between FDM carriers for interference prevention.

Orthogonal Frequency-Division Multiple Access (OFDMA) is an accessing scheme based on the OFDM technique, and is widely applied in the current 4G *Long Term Evolution* (LTE) networks. It is the multi-user version of OFDM. By further dividing carriers into subcarriers and allocating subcarriers to different users, multiple access is achieved. While OFDMA may assign an individual user more than one subcarrier, a subcarrier can only be assigned to one single user. The allocation will vary depending on the number of users in the system, and the demand of each user.

Figure 1.6 shows subcarrier allocation changes over time. We can see that, at the initial time slot, 8 subcarriers were evenly allocated to User 0 and User 1. However,

the two users left the system in the next time slot and all subcarriers were allocated to User 2. The adaptive subcarrier allocation in OFDMA systems is recognised as an efficient multiple access approach for wireless networks.

All four papers in this thesis are based on the OFDMA scheme. The subcarrier and power allocation problem in OFDMA systems will be introduced in subsection 1.1.2–1.1.4.

1.1.2 Single-user OFDMA systems

Data are transmitted over subcarriers in OFDMA networks and there are many factors which affect the transmission rate. The commonly known Shannon-Hartley theorem in information theory tells the capacity of a subcarrier (i.e. theoretical maximum bit-rate that can be transmitted, see [45]) given its bandwidth and noise. It states that:

$$C = B\, log_2(1 + \frac{p}{N})\ bits/second, \tag{1.1}$$

where C is subcarrier capacity, B is subcarrier bandwidth measured in Hertz, p is power in Watts supplied to the subcarrier and N is noise in Watts on that subcarrier.

For a given subcarrier, its bandwidth B is usually a known constant. Noise on subcarriers varies from time to time, but for simplification we use the average noise of that subcarrier, so that we will have a non-linear function of C against p.

Consider an OFDMA system where there is one user and 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:

$$\max \left\{ \sum_{i \in I} f_i(p_i) :\ \sum_{i \in I} p_i \leq P,\ p \in \mathbb{R}_+^{|I|} \right\}. \tag{1.2}$$

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 [8]). It can also be solved by a specialised iterative technique called *water filling*; see, e.g., [12, 27].The idea behind water-filling is to view the individual subcarriers as containers and the available power as water that is to be poured into those containers. Subcarriers with higher signal-to-noise ratio are viewed as larger containers. The algorithm starts with all containers empty, and then iteratively pours water into

non-full containers at a constant rate. From a formal point of view, water-filling can be regarded as a variant of the steepest-ascent (a.k.a. gradient or hill-climbing) method. For brevity, we do not introduce details of the algorithm, but one can find a comprehensive example at [1].

1.1.3 Multi-user OFDMA systems

As mentioned in subsection 1.1.1, OFDMA systems are multi-user systems, and each subcarrier can be assigned to an arbitrary user. Now we discuss problem 1.2 in a multi-user scenario.

Suppose we have subcarrier set I and user set J, a bandwidth $B_i > 0$ and noise $N_i > 0$ for each $i \in I$, and a power limit $P > 0$. 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.

We formulate the problem as an MINLP (see 1.2.4 for an introduction). 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:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} f_i(p_{ij}) && && (1.3)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P && && (1.4)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 && (\forall i \in I) && (1.5)\\
& p_{ij} \le P x_{ij} && (\forall i \in I,\ j \in J) && (1.6)\\
& p_{ij} \in \mathbb{R}_+ && (\forall i \in I,\ j \in J) && (1.7)\\
& x_{ij} \in \{0,1\} && (\forall i \in I,\ j \in J). && (1.8)
\end{aligned}$$

The objective function (1.3) represents the total data rate. The constraint (1.4) imposes the power limit. The constraints (1.5) ensure that each subcarrier is allocated to one user at most. The constraints (1.6), 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.

Although this problem is an MINLP, it can be solved easily. Indeed, all we have to do is solve the problem (1.2), and then assign each subcarrier to an arbitrary user. In practice, however, one often encounters more complex variants of this basic problem, which are much harder to solve in both theory and practice. We review the main problem variants in the next subsection.

1.1.4 More complex multi-user problemss

The first problem that we focus on in this thesis 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) problem.

If we let J denote the set of users, then the total data rate for that user will be $\sum_{i\in I} f_i(p_i)$. The constraint 1.6 ensures that $p_{ij} = 0$ when $i \notin S_j$, therefore $\sum_{i\in I} f_i(p_i) = \sum_{i\in S_j} f_i(p_i)$. We add an extra constraint to the multi-user OFDMA problem so that data rate for user j should be at least the demand of user j, which we denote by ℓ_j:

$$\sum_{i\in I} f_i(p_{ij}) \geq \ell_j \qquad (\forall j \in J). \tag{1.9}$$

There are many papers on such optimisation problems in OFDMA systems. Wong & Cheng [82] minimise total power subject to individual quality of service (QoS) constraints that impose a lower bound on the data rate for each user. Rhee & Cioffi [65] 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.* [41] consider the problem of maximising total data rate subject to a total power limit. Shen *et al.* [73] add a global QoS constraint to that problem. Seung *et al.* [70] enforce QoS by giving each user a weight, and maximising a weighted sum of the data rates. Yu & Lui [87] consider an extension of the problem in [70], in which there is interference between channels. Tao *et al.* [76] take the problem in [41], 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., [23, 34, 79, 84, 85, 88]). Therefore, the second problem we will introduce in this thesis is the *fractional subcarrier and power allocation problem with rate constraints* (F-SPARC) problem, continuing this trend. The F-SPARC problem is very similar to the SPARC problem, but with objective function being maximising *energy efficiency*, which is defined as total data rate divided by total power supplied to the system:

$$\max \frac{\sum_{i\in I}\sum_{j\in J} f_i(p_{ij})}{\sigma + \sum_{i\in I}\sum_{j\in J} p_{ij}}. \tag{1.10}$$

Most of the aforementioned problems are proved to be $\mathcal{NP}$-hard in the strong sense (see [32, 52, 53]), we conjecture that both the SPARC and F-SPARC problems

are $\mathcal{NP}$-hard in the strong sense as well. However, the function f_i is concave over the domain $[0, P]$ for all $i \in I$. As a result, the objective function (1.3) is concave. This means that the problem is convex, therefore, its continuous relaxation can be solved efficiently via convex programming techniques.

1.2 Optimisation

Optimisation, also known as *mathematical programming*, is a discipline branched from applied mathematics, the goal of which is finding the *best* solutions for real-world problems using mathematical tools.

Optimisation has been widely applied in a number of fields including transportation, manufacturing, inventory control, scheduling, and networking. Developments in telecommunications has also provided a rich source of optimisation problems, including our SPARC and F-SPARC problems (see [64]).

A typical optimisation problem often consists of an *objective function*, *variables* and various *constraints*. *Solving* an optimisation problem means finding values of the variables that maximise or minimise the objective function, subject to all constraints. A basic optimisation problem takes the form

$$\begin{array}{ll} \max & f_0(x) \\ s.t. & f_i(x) \leq 0 \quad i = \{1, ..., m\}, \\ & x \in \mathcal{X} \end{array} \tag{1.11}$$

where x is the variable vector, $\mathcal{X}$ is the domain of the variables (continuous, integer, binary, or some mixture), function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and functions $f_i : \mathbb{R}^n \to \mathbb{R},\ i = \{1, ..., m\}$ are the constraints. A solution is feasible if it satisfies all m of the constraints, and the area formed by all feasible solutions is called the feasible region. If the problem is to minimise the objective function, one can easily multiply the objective function by -1. Equation constraints can also be added to an optimisation problem, but they are not covered in this thesis.

In this section, we will introduce common optimisation problems involved in this paper.

1.2.1 Linear programming

The most fundamental optimisation problem is the *linear programming* (LP) problem, which takes the form

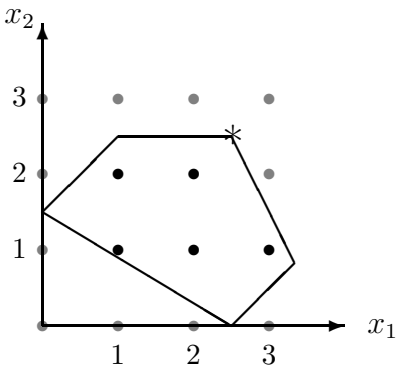


Figure 1.7: Feasible region of LP example bounded by five constraints

$$\begin{array}{lrcl} \max & c^T x & & \\ s.t. & Ax & \leq & b \\ & x & \geq & 0, \end{array} \tag{1.12}$$

where A is a matrix of constants, c and b are column vectors, x is the variable vector, b and c are vectors of given coefficients.

We show an example of a maximisation LP problem here with its feasible region plotted on a 2D graph shown in figure 1.7:

$$\begin{array}{lrcrcr} \max & x_1 & + & x_2 & & \\ s.t. & -2x_1 & + & 2x_2 & \leq & 3 \\ & 2x_1 & - & 2x_2 & \leq & 5 \\ & -6x_1 & - & 10x_2 & \leq & -15 \\ & & & 2x_2 & \leq & 5 \\ & 4x_1 & + & 2x_2 & \leq & 15 \\ & x_1, & x_2 & & \geq & 0 \end{array} \tag{1.13}$$

It is proven in [60] that if a linear programming problem has one or more points which maximises (or minimises in minimisation problems) the objective function, then at least one of those points is the extreme point. It is also shown that if an extreme point is not the optimal solution, there must be an edge which has a better solution on the other side connected to this point. Solutions on extreme points are called corner point solutions. In the case of problem (1.13), the corner point solutions are $\{(0, 1.5), (1, 2.5), (2.5, 0), (\frac{1}{3}, \frac{5}{6}), (2.5, 2.5)\}$, and the optimal solution is $(2.5, 2.5)$. A very effective algorithm called the *simplex method* was proposed by Dantzig in 1947 to solve linear programming problems by searching corner point solutions.

The simplex method first finds a corner point solution by solving a system of simultaneous linear equations, and then checks whether that solution is optimal. If it

is not, then it searches all edges connected to this point to find the one which offers the best corner point solution and moves to that extreme point. The algorithm stops once the optimal solution has been found.

Unfortunately, simplex method is not a polynomial-time algorithm (see [43]), and can encounter difficulties when solving large-scale LP problems. A few techniques has been proposed for those LPs.

Benders decomposition, also known as *row generation*, is a technique for solving LP problems with block structures (see [4,14]). Firstly, it divides all variables into two subsets, and solves a master problem based on one subset of variables. Some variable values of the optimal solution from this stage will be fixed and a sub-problem will be formulated with the other subset of variables. For those given fixed variable values, if the subproblem is infeasible, *Benders cut* will be added to be master problem in order to reduce searching space. The process will continue until no Benders cuts can be generated.

Column generation is another technique widely used to solve LP problems with a large number of variables (see [14]). A master problem is derived from the original problem and only considers a subset of variables. A subproblem is then created to identify a new variable to add to the master problem, utilising reduced cost in duality theory. The algorithm keeps adding new variables when its reduced cost is negative and will stop when no variables can be added to the master problem.

Several other algorithms have been proven to run in polynomial time (e.g., Khatchian's ellipsoid method in [40], and the Karmarkar's projective method in [36]). Therefore, in computational complexity theory, LP problems are classified as $\mathcal{P}$ problems, which means they can be solved in polynomial time, such that the time to solve the problem increases as a polynomial function as the size of the input increases. An LP problem of reasonable scale is often easy to solve by most solvers available.

1.2.2 Non-linear programming

In contrast to linear programming, a problem is a *non-linear programming* (NLP) problem if the objective function or any of the constraints in problem (1.11) is not linear.

Non-linear programming problems are generally much more difficult to solve than LP problems. For some NLP problems, there are no known algorithms to solve in polynomial time, but if a solution is provided, it is possible to verify the solution in polynomial time. Those problems are classified as $\mathcal{NP}$ problems. However, most NLP problems are in the class of $\mathcal{NP}$-hard, meaning they are at least as complex as

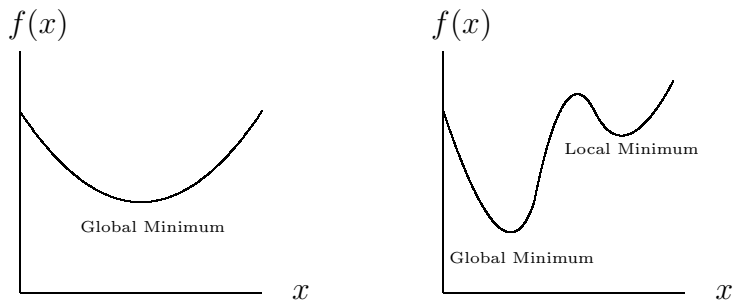


Figure 1.8: A convex function (left) and a non-convex one (right)

the most difficult $\mathcal{NP}$ problems ($\mathcal{NP}$-complete problems). Although not yet proven, it is widely believed that there are no polynomial-time algorithms for $\mathcal{NP}$-complete problems.

One important type of NLP optimisation problem is the *convex optimisation* problem. While convex optimisation is now a somewhat mature technique, it has been proven in [61] that a non-linear programme with a non-convex quadratic function is $\mathcal{NP}$-hard even when the only constraints present are non-negativity constraints.

We now present definitions related to convex optimisation before we can explain further.

Definition 1 *A function $f : X \to \mathbb{R}$ is convex if:*

$$f\big(tx_1 + (1-t)x_2\big) \leq tf(x_1) + (1-t)f(x_2) \qquad \forall x \in X, \forall t \in [0,1].$$

Definition 2 *A function $f : X \to \mathbb{R}$ is concave if -f is convex.*

Definition 3 *A set C is convex if for any two points $x_1, x_2 \in C$ and any θ in $[0,1]$, we have:*

$$\theta x_1 + (1-\theta)x_2 \in C.$$

Definition 4 *A problem is a convex optimisation problem if it takes the form* (1.11) *and functions $f_i,\ i = \{0, ..., m\}$ are all* convex functions.

Definition 5 *A point x^* is a global maximum of f if $f(x^*) \geq f(x)\ \forall x \in S$, where S is the feasible region.*

The property which makes convex optimisation problems easier to solve than non-convex optimisation problems is that any one of its *local optima* is also the *global optimum* (See Figure 1.8). Global optimum is referred to as the truly optimal solution

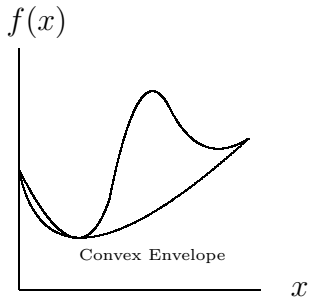


Figure 1.9: Convex envelope of a non-convex function

over the whole feasible region, whereas local optima are the best solutions among a subset of the feasible region. Therefore, when minimising a non-convex objective function f, one can utilise a convex function which underestimates or equals f in the same domain. The convex function is the convex envelope of the non-convex function f. On the other hand, concave envelopes are applicable in maximisation problems (see Figure 1.9).

An *Outer approximation* (OA) algorithm was designed for convex NLPs, proposed by Kelley (see [38]) in 1960s. Suppose we have a convex NLP of the following form:

$$\max\left\{f(y):\ y \in \mathcal{C} \subseteq \mathbb{R}^n_+\right\}, \tag{1.14}$$

where $f(y)$ is a convex non-linear function of the decision variable vector y, and $\mathcal{C}$ is the domain of y.

The problem can be reformulated into:

$$\max\left\{z:\ z \leq f(y),\ z \in \mathbb{R},\ y \in \mathcal{C} \subseteq \mathbb{R}^n_+\right\} \tag{1.15}$$

by introducing an extra continuous variable z.

The key idea of Kelley's algorithm is to approximate the convex constraint with a collection of linear constraints of the form:

$$z \ \leq\ f(\bar{y}) + f'(\bar{y})\left(y\ -\ \bar{y}\right), \tag{1.16}$$

where f' denotes the first derivative of f. The constraints (1.16) are called *Kelley cuts*. By approximating all non-linear constraints with Kelley cuts, the problem is then an LP relaxation of the original NLP. The algorithm converges to a solution with acceptable optimality gap when more and more Kelley cuts are added.

There have been some quite effective solvers (e.g.: Mosek, Ipopt, CVXOPT etc.) for convex optimisation problems.

1.2.3 Mixed integer linear programming

In real life there are always problems whereby solution spaces are discrete rather than continuous. Problems of this type are called *discrete optimisation* which usually, but not necessarily, involves integral variables. A particularly important application of discrete optimisation is to model and solve *combinatorial optimisation problems* (see [44]). These are optimisation problems that involving concepts from combinatorics (such as sets, subsets, combinations and permutations) and graph theory (such as nodes, edges, cliques, cuts and flows).

Most *combinatorial optimisation* problems can be modelled as *Mixed integer programmings* (MIPs) or *Mixed integer linear programmings* (MILPs),in which some variables can only take integral values. MILP is itself a subset of discrete optimisation, and takes the form of

$$\begin{array}{rrcl} \max & c^T x & + & d^T y \\ s.t. & Ax & + & Ey \quad \leq \quad b \\ & x, y & \geq & 0 \\ & y & \in & \mathbb{Z}. \end{array} \tag{1.17}$$

When x is an empty set, i.e., all variables are required to be integers, the problem is often called *integer programming* (IP). For brevity, we mention both types of programmes as MIP below.

A special case of an MIP problem is called *binary integer programming*, where some, or all, integral variables can only take the value 0 or 1. A *binary variable* is often considered as a "yes-or-no" variable: it takes value 1 when the answer is "yes" and 0 for "no".

We continue using the LP example in subsection 1.2.1 with an extra constraint that both x_1 and x_2 are integers, so it becomes an IP problem. The feasible region of the problem is then discrete due to the new integral constraint. As shown in Figure 1.10 as black dots, there are five integral solutions bounded by the five constraints.

Although MIP is similar in form to LP, the involvement of integrity of variables makes the difficulty of an optimisation problem increase exponentially. IP problems are proven to be $\mathcal{NP}$-hard in general (see [37]). Since MILP problems are even more general than ILP, they also tend to be $\mathcal{NP}$-hard. Many efforts have been made to solve MIP problems, and we will introduce some of the most successful algorithms in the remaining part of this subsection.

One of the most successful MIP techniques is the *cutting plane* method put forward by Gomory in 1958 (see [24]). The cutting plane method generates and adds

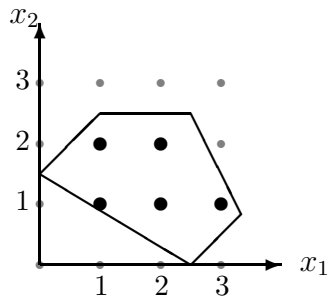


Figure 1.10: Feasible region of IP example bounded by five constraints

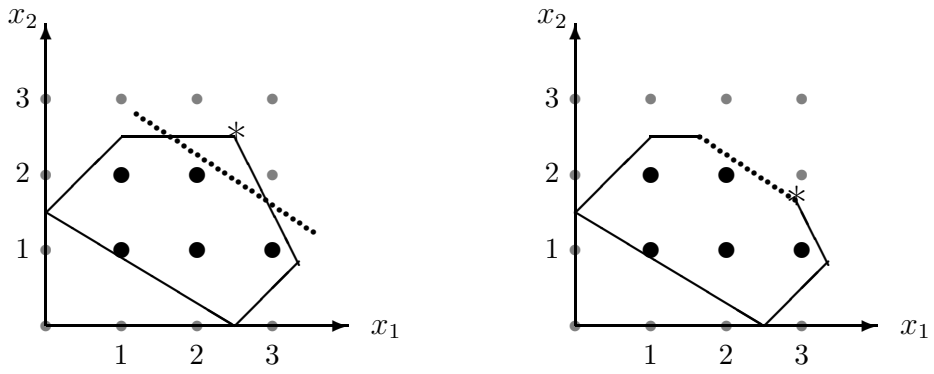


Figure 1.11: Cutting plane method for integer programming

new linear, non-redundant constraints to the original problem in order to reduce the size of the feasible region.

Geometrically, a cutting plane is a hyperplane which separates all MIP feasible solutions on one of its sides (see Figure 1.11). A cutting plane (or cut) is only valid if it eliminates some of the feasible region and is satisfied by all feasible solutions of the original MIP problem.

By keeping added cutting planes, the optimal integral solution will eventually become an extreme point and can be obtained by solving the relaxation problem. It is then critical to find strong cutting planes in order to cut off as much as possible. We show two types of strong cuts in Figure 1.12. The cutting plane on the left touches the polyhedron formed by all integral solutions, while the one on the right defines a facet of the polyhedron and is the strongest possible cutting plane.

The cutting plane method is the first algorithm proposed for MIP which was proven to converge. Although it is not considered efficient, it has inspired many further algorithms.

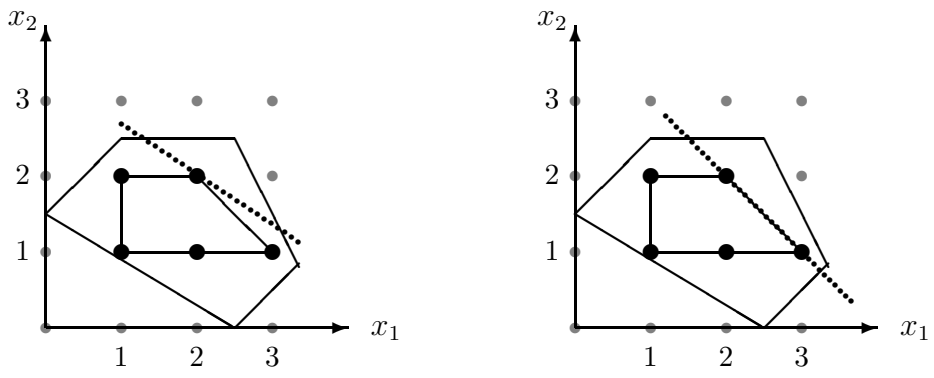


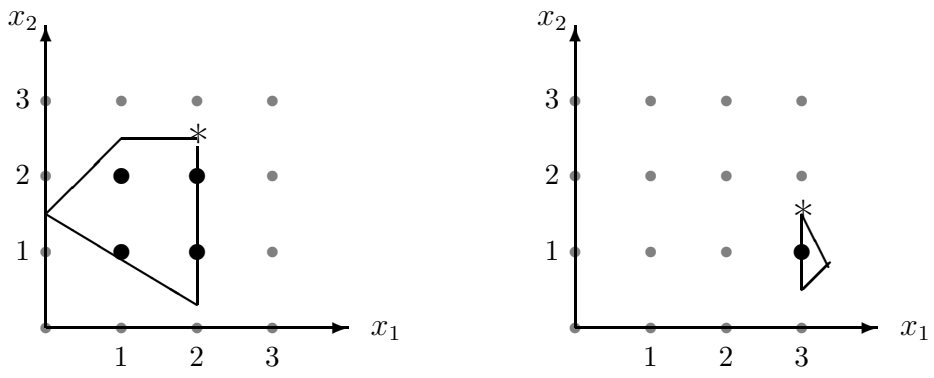
Figure 1.12: Strong cutting plane method for integer programming

Another successful method is the *branch and bound* (BB) method (see [46]). The branch and bound method partitions the solution space of a problem into a tree structure, and explores the solution space node by node until it finds the optimal solution.

First, the branch and bound algorithm solves the relaxation of the original MIP maximisation problem to get an upper bound, it then obtains a lower bound based on the relaxed solution. Take the IP problem as an example, we know from subsection 1.2.1 that the optimal objective value is 5 when $x_1 = x_2 = 2.5$. However, this solution is fractional, thus infeasible. However, the relaxed optimal value can be used as an upper bound, since no solution of the original problem can yield an objective value higher than 5. We can then round-down the two variables to 2, in order to obtain a lower bound of this problem.

The algorithm then partitions the solution space based on a variable by generating two sub-problems. Suppose we choose variable x_1 to partition the solution space. We get two natural sub-problems, in each of which there exists an additional constraint. In one of the sub-problems, the additional constraint is $x_1 - 2 \leq 0$; the one in the other sub-problem is $x_1 - 3 \geq 0$. The two sub-problems are presented in Figure 1.2.3. By solving the first sub-problem, we get a relaxed optimal solution $(2, 2.5)$. In the second sub-problem, the relaxed optimal solution is $(3, 1.5)$.

The branch and bound keeps a record of upper bounds and lower bounds of all nodes. When the upper bound of a node is smaller or equal to the lower bound of another node, the former node will be eliminated. The algorithm keeps partitioning generated nodes until the best lower bound found is the same as the highest upper bound in un-eliminated nodes, or all nodes have been explored. A very detailed and comprehensive branch and bound example can be found in [78].



The branch and bound method is intelligent in four aspects:

1. The enumeration process stops when optimal solution is found
2. It recognises solution space where no optimal solution exists
3. It recognises solution space where no feasible solution exists
4. It searches the most promising solution space first

The branch and bound method is an exact algorithm which means it is guaranteed to find the optimal solution. However, it is an enumeration algorithm, it needs to solve a huge number of sub-problems when facing complex problems, and that can cause memory and time issues.

We remark that both Bendess Decomposition and Column Generation can be applied to MILPs. By combining those algorithms with B&B, branch and cut (see [62]) and branch and price (see [2]) are proposed and widely used for solving large MILPs.

1.2.4 Mixed integer non-linear programming

The most difficult class of optimisation problem is formed by the *mixed integer non-linear programming problems* (MINLP), which take the form

$$\begin{array}{lrcl} \min & f(x,y) & & \\ s.t. & c(x,y) & \leq & 0 \\ & x & \in & \mathbb{R} \\ & y & \in & \mathbb{Z} \end{array} \tag{1.18}$$

where f and c are twice continuously differentiable functions, x and y are the vectors of continuous variables and integer variables respectively. If functions f and c are both convex, the problem is called *convex MINLP*, otherwise *non-convex MINLP*.

MINLP are optimisation problems which consist of both non-linear elements (objective function and/or constraints) and integral variables. As a result, most MINLP problems are $\mathcal{NP}$-hard.

Solving an MINLP problem is usually very challenging due to the fact that it combines the difficulties from both NLP and MIP problems. We now introduce some of the algorithms available for MINLP problems.

1. Branch and bound can also be applied to MINLPs. The non-linear BB method solves an NLP instead of an LP relaxation problem during its iterative routine, it does not differ greatly from the linear BB method. However, as mentioned in subsection 1.2.3, the BB algorithm is not efficient when there are too many sub-problems to solve.

2. Outer Approximation algorithms can also be used to solve convex MINLP problems. By solving two sub-problems derived from the original MINLP problem, the OA method obtains an upper bound and a lower bound. An MILP master problem is then formulated, and cutting planes are added to it based on the relaxation solution obtained from one of the sub-problems. The MILP master problem is a relaxation of the original MINLP problem, but by adding cutting planes effectively, it will approximate the non-linear feasible region and update bounds in every iteration. The OA algorithm stops when the difference of the two bounds becomes smaller than some criterion. Indeed, Kelley's cuts can be integrated with Gomory's cuts to solve convex MINLPs.

3. *Generalised Benders decomposition* (GBD) is generalised from the Benders decomposition method to solve convex MINLP problems. The GBD method does not require sub-problems to be linear, but some issues have been found: in some cases it may only converge to a local minima, in other cases it may not converge at all.

4. *LP/NLP based branch and bound* is generally similar to branch and bound but avoids solving expensive MILP relaxations. In each node of tree search in branch and bound, if an integer solution is found, an NLP sub-problem with the fixed integer solution values will be solved by adding cutting planes to the single tree node. This method is often suited for convex MINLP problems where MILP relaxations are too difficult to solve.

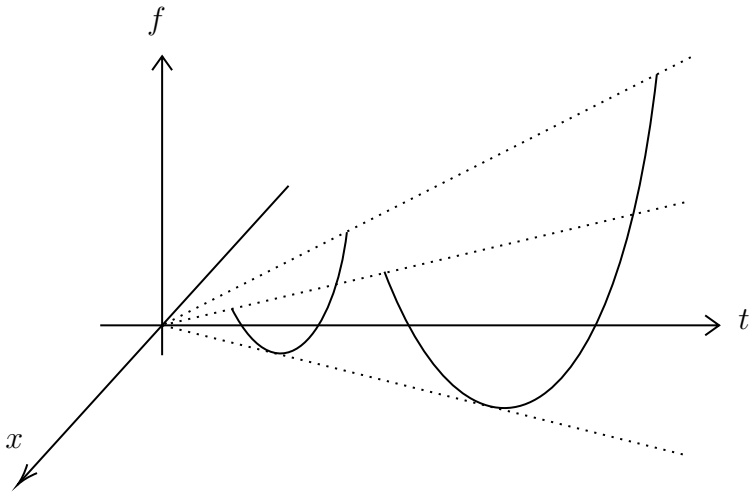


Figure 1.13: Example of a perspective function

MINLP problems are extremely difficult to solve. The OA, GBD and LP/NLP based BB methods are all for convex MINLP problems only. For non-convex MINLP problem, a comprehensive list of solvers can be found at [57]. There are some good surveys available for both convex and non-convex MINLP problems, see [6, 9, 13, 19, 25, 29, 51].

1.2.5 Perspective Function and Perspective Cuts

A very importance concept we will use in this thesis is the perspective function (see [31]). The perspective of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the function $g : \mathbb{R}^{n+1} \to \mathbb{R}$, defined by $g(x, t) = tf(x/t)$. By definition, the perspective function g takes the value 0 when $t = 0$. The perspective operation of a function preserves convexity (concavity) of the original function f.

Obviously, the perspective function g is equivalent to the original function f when $t = 1$. We show the perspective of f in Figure 1.13. One can see that the variable t only scales the function f but has not effect on its shape.

One of the important applications of perspective functions is to solve *fractional programs*, which takes the form:

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

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 the domain C. It is shown in [10, 69] that by utilising perspective function, such a problem can be reformulated as

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

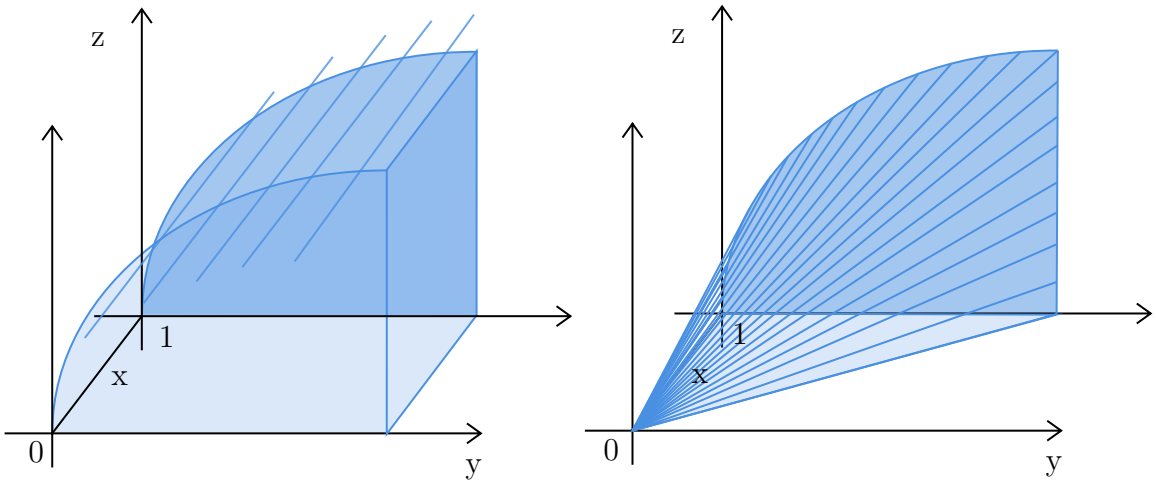


Figure 1.14: Perspective cuts of a nonlinear function with an indicator variable

where t is a new non-negative variable representing $1/g(y)$, and y' is a new vector of variables representing $y/g(y)$. The reformulated problem can often be solved efficiently, since the objective function is concave and the feasible region is convex.

Frangione & Gentile also showed that perspective functions can be used to generate stronger cutting planes when solving MINLPs with indicator variables.

We now introduce an indicator variable x to the problem (1.14) so that if x takes the value zero, then all of the components of y must also take the value zero. The problem then becomes an MINLP as following:

$$\max\left\{f(y):\ y_j \leq Mx,\ x \in \{0,1\}, y_j \in \mathcal{C} \subseteq \mathbb{R}^n_+\ \forall j \in \{1,...,n\}\right\}, \tag{1.19}$$

where M is the largest values y can take.

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* (see [22]). Note that, when $x = 1$, they reduce to Kelley cuts, but when $x < 1$, they are stronger.

Geometrically, the original Kelley cuts are a collection of planes over the feasible region of the NLP relaxation of problem (1.19), whereas the strengthened ones are planes cutting thought that feasible region connecting the non-linear function f and the grid origin so that the non-linear function f is projected into a point as x reduces from 1 to 0 (see Figure 1.14). The strengthened cuts are very helpful when solving the MILP or even LP relaxation of problem (1.19).

When dealing with a *fractional program* with indicator variables, we found that, in order to obtain tight convex relaxations of such problems, perspective cuts can also be applied, but one needs to study a new kind of function, which we call a *bi-perspective* (Bi-P) function.

Let y be a vector of n continuous variables, let $f(y)$ be a real function of y that is defined over a convex domain $C \subseteq \mathbb{R}^n_+$, and let t_1 and t_2 be non-negative continuous variables. The Bi-P function of $f(y)$ is defined as:

$$t_1\, t_2\, f\left(\frac{y}{t_1\, t_2}\right),$$

with domain $t_1, t_2 \in \mathbb{R}_+$, $y \in t_1\, t_2\, C$. (By convention, the Bi-P function takes the value zero when $t_1\, t_2 = 0$ and y is the origin.)

Whereas standard perspective functions preserve convexity and/or concavity, the same is not true for Bi-P functions. So we need to obtain the concave envelope of the Bi-P function as follows.

Define the following two auxiliary functions:

$$\begin{aligned} g^1(t_1, t_2) &= a_2 t_1 + b_1(t_2 - a_2) \\ g^2(t_1, t_2) &= a_1 t_2 + b_2(t_1 - a_1). \end{aligned}$$

We prove in Chapter 3 that the concave envelope, over the domain $t_1 \in [a_1, b_1]$, $t_2 \in [a_2, b_2]$ and $y \in t_1 t_2 C$, is:

$$\min\left\{ g^1(t_1, t_2)\, f\left(\frac{y}{g^1(t_1, t_2)}\right),\ g^2(t_1, t_2)\, f\left(\frac{y}{g^2(t_1, t_2)}\right)\right\}. \tag{1.20}$$

We then obtain the hypograph of the function (1.20) described by the linear inequalities

$$z \ \leq\ \nabla f(\bar{y})\, y \ +\ \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, g^k\big(t_1, t_2\big) \tag{1.21}$$

for $\bar{y} \in C$ and $k = 1, 2$. We call the inequalities the Bi-P cuts.

We remark that when t_2 is an indicator variable, the Bi-P cuts reduce to:

$$z \ \leq\ \nabla f(\bar{y})\, y \ +\ \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, b_1 t_2 \tag{1.22}$$

$$z \ \leq\ \nabla f(\bar{y})\, y \ +\ \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, \Big(t_1 - a_1(1 - t_2)\Big). \tag{1.23}$$

1.3 Research Contribution

1.3.1 Research questions

As mentioned in subsection 1.1.4, a lot of the SPARC problem variants considered in the cited papers are $\mathcal{NP}$-hard in the strong sense. Accordingly, in all of the those papers, the authors use relaxation techniques to compute bounds, and heuristics to find feasible solutions. An issue arises in whether we can solve the SPARC problems exactly using MINLP.

While exact algorithms put emphasis on optimality, we are also concerned with heuristics that can solve SPARC problems both effectively and efficiently in realistic scenarios. Moreover, we are also interested in a heuristic for dynamic and overloaded systems, where total user demand changes frequently and is sometimes higher than system capacity.

1.3.2 Methodology

As a kind of natural science, optimisation has had a somewhat standardised methodology since its inception in the 1950s when researchers started Operational Research. Since then, the methodology used for optimisation has remained largely unchanged. Approaches for optimisation follow specific steps, and for each step there corresponds a specific methodology. Almost all researches in optimisation follow the specific steps presented in this subsection.

1. Understanding and describing problem
2. Building a mathematical model
3. Development and refinement of the model
4. Algorithm design
5. Algorithmic complexity analysis
6. Coding and debugging
7. Design of test instances
8. Computational experiments
9. Solution analysis

1.3.3 Overview of papers

We now present the structure of this thesis in this subsection.

In Chapter 2, we will introduce the SPARC problem and its variants in subsection 2.3. We will then present an exact algorithm in subsection 2.4, which uses a combination of outer approximation [18] and perspective cuts [22, 26], together with three implementation "tricks" of our own, which improve the running time by several orders of magnitude. Computational experiment results given in subsection 2.5 show that our algorithm is capable of solving SPARC problem instances of realistic size and complexity, to proven optimality, in about one minute.

Following on, we will introduce an exact algorithm for the F–SPARC problem in Chapter 3. In subsection 3.3, we will define bi–perspective (BP) function and characterise the concave envelope of a BP function over a rectangular domain. We then derive a family of linear inequalities, which we call *BP cuts*, that completely describe the concave envelope. In subsection 3.4, we will demonstrate how to generalise the BP cuts when there are "multiple-choice" constraints, stating that two or more indicator variables cannot take the value 1 simultaneously. Computational experiment results in subsection 3.6 show that the new cuts typically close over 95% of the integrality gap.

In Chapter 4, we will present a heuristic algorithm which solves the same problem as in Chapter 3. We then consider the SPARC problem in dynamic and overloaded cases in Chapter 5. In subsection 5.3.2 we introduce an objective function which takes fairness into consideration. A dynamic local search heuristic is then proposed in subsection 5.4.

Finally, in Chapter 6, we conclude the thesis and give suggestions for future work.

Chapter 2

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

2.1 Introduction

Telecommunications provides a rich source of interesting, and often challenging, optimisation problems (see, e.g., Resende & Pardalos [64]). 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 [17]).

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 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., [23, 41, 52, 53, 65, 70, 73, 76, 79, 82, 84, 85, 87, 88]). 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 [18] and perspective cuts [22, 26], 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 2.2. In Section 2.3, we define the SPARC formally and present a convex MINLP formulation for it. The exact algorithm is described in Section 2.4. The results of some extensive computational experiments are given in Section 2.5, and some concluding remarks are made in Section 2.6.

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., [3, 13, 81].

2.2 Literature Review

Now we briefly review the literature on optimisation in OFDMA and related systems. Good reference texts are [12, 17, 27].

2.2.1 Single-user systems

First, consider a single communications channel and a single user. The classical *Shannon–Hartley theorem* [71] 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:

$$\max \left\{ \sum_{i \in I} f_i(p_i) : \sum_{i \in I} p_i \leq P,\ p \in \mathbb{R}_+^{|I|} \right\}. \tag{2.1}$$

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 [8]). It can also be solved quickly by a specialised iterative technique called *water filling*; see, e.g., [12, 27].

2.2.2 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 [82] 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 [65] 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.* [41] consider the problem of maximising total data rate subject to a total power limit. Shen *et al.* [73] add a global QoS constraint to that problem. Seung *et al.* [70] enforce QoS by giving each user a weight, and maximising a weighted sum of the data rates. Yu & Lui [87] consider an extension of the problem in [70], in which there is interference between channels. Tao *et al.* [76] take the problem in [41], 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., [23, 34, 79, 84, 85, 88]).

It is proved in [52, 53] 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.

2.3 Problem Definition and MINLP Formulations

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

2.3.1 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 [41, 73], 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 [76,82], 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.

2.3.2 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:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} f_i(p_{ij}) && && (2.2)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P && && (2.3)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 && (\forall i \in I) && (2.4)\\
& \textstyle\sum_{i\in I} f_i(p_{ij}) \ge \ell_j && (\forall j \in J) && (2.5)\\
& p_{ij} \le P x_{ij} && (\forall i \in I,\, j \in J) && (2.6)\\
& p_{ij} \in \mathbb{R}_+ && (\forall i \in I,\, j \in J) && (2.7)\\
& x_{ij} \in \{0,1\} && (\forall i \in I,\, j \in J). && (2.8)
\end{aligned}$$

The objective function (2.2) represents the total data rate. The constraint (2.3) imposes the power limit. The constraints (2.4) ensure that each subcarrier is allocated to at most one user. The constraints (2.5) are the user rate constraints. The constraints (2.6), 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 (2.2) is concave, and the constraints (2.5) 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 [70, 87], then one changes the objective function (2.2) 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 [34], one changes P to u in the VUBs (2.6). 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 (2.5) to $\ell_j z_j$, where z_j is a new binary variable, and changes the objective function to $\sum_{j\in J} z_j$.

2.3.3 Modified convex MINLP formulation

It is well known (see, e.g., Section 2 of [77]) 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:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} r_{ij} && && (2.9)\\
\text{s.t.} \quad & (2.3), (2.4), (2.6) - (2.8) && && (2.10)\\
& \textstyle\sum_{i\in I} r_{ij} \geq \ell_j && (\forall j \in J) && (2.11)\\
& r_{ij} \leq f_i(p_{ij}) && (\forall i \in I, j \in J). && (2.12)\\
& r_{ij} \in \mathbb{R}_+ && (\forall i \in I, j \in J). && (2.13)
\end{aligned}$$

With these modifications, eveything is linear apart from the (convex) constraints (2.12).

2.4 An Exact Algorithm for the SPARC

We now present our exact algorithm for the SPARC. Subsection 2.4.1 presents the algorithm in its simplest form. Enhancements to the algorithm are presented in Subsections 2.4.2 to 2.4.5.

2.4.1 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 [15,18]. The key idea is to approximate the convex constraints (2.12) with a collection of linear constraints of the form:

$$r_{ij} \;\leq\; f_i(\bar{p}) + f_i'(\bar{p})\,\big(p_{ij} \;-\; \bar{p}\big), \tag{2.14}$$

where the $\bar{p}$ values are selected from the domain $[0, P]$, and f_i' denotes the first derivative of f_i. The constraints (2.14) are called *Kelley cuts*, since they were first developed by Kelley [38] 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 (2.12), and the parameter ϵ_2 is the maximum acceptable relative gap between the final values of the bounds U and L.

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 (2.12) is violated by more than* ϵ_1 **then**
      Let $\bar{p} = f^{-1}(r^*_{ij})$;
      Generate the Kelley cut (2.14) 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.

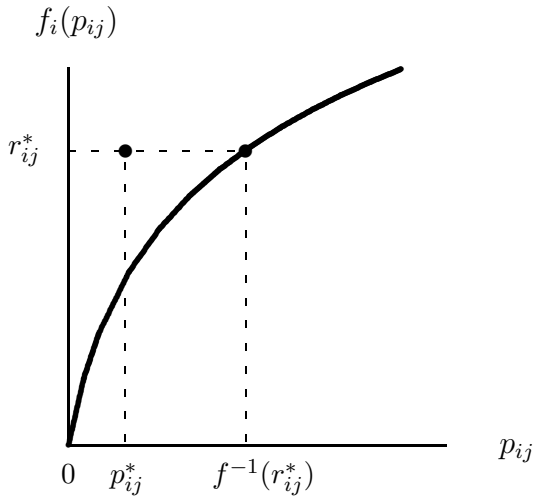


Figure 2.1: Choosing $\bar{p}$ in order to avoid numerical issues.

- 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 2.1).

- To construct our initial MILP relaxation, we include the objective function (2.9), the constraints (2.3), (2.4), (2.6)-(2.8), (2.11) and (2.13), and a collection of $|I|\,|J|$ Kelley cuts; namely, those Kelley cuts that are tight at the optimal solution to the continuous relaxation of (2.9)–(2.13).

- 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.

2.4.2 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 [22].

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 (2.14) with the perspective cuts

$$r_{ij} \;\leq\; f_i'(\bar{p})\, p_{ij} \;+\; \Big(f_i(\bar{p}) - f_i'(\bar{p})\, \bar{p} \Big)\, x_{ij} \quad (\forall i \in I,\, j \in J,\, \bar{p} \in [0, P]). \tag{2.15}$$

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.* [20, 21]. Applications in location, scheduling, network design, finance and power generation are surveyed in Günlük & Linderoth [26].

2.4.3 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 [54] 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 $|J|$ 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 (2.14) for the given i, j and $\bar{p}$ "

with the line

"For all $j \in J$, generate the perspective cut (2.15) 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 2.2 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.

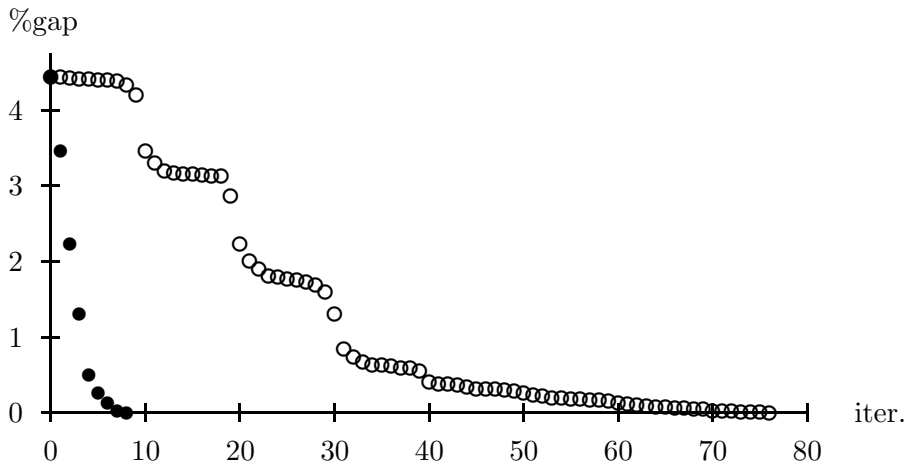


Figure 2.2: Typical evolution of percentage gap between upper bound and optimum, without pre-emptive cut generation (hollow circles) and with pre-emptive cut generation (filled circles).

2.4.4 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 (2.1). 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 subcarriers 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:

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

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 2.5.)

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 2.5 show that, in fact, the range of ℓ_j values for which pre-processing fails is very narrow.

2.4.5 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 (2.9)–(2.13), 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 2.4.1). 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.

2.5 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 (2.1) 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%.

2.5.1 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 2.4.4), 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

Table 2.1: 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

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 (2.1).

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 (2.1) 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.

2.5.2 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 2.1 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.

Table 2.2: 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

Table 2.3: 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

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.

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 2.2, 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 2.3, 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 $|J|$.

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).

2.5.3 Comparison with BONMIN

Finally, we compared our algorithm with the open-source MINLP solver BONMIN [5]. In our initial experiments, we tried feeding four different formulations to BONMIN: the original formulation (2.2)–(2.8), the modified formulation (2.9)–(2.13), 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 subcarriers and users, we created instances with DR set to 20%, 50% and 99%. This makes 12 instances in total.

Table 2.4 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 2.4.1, and (ii) the high degree of near-symmetry, as discussed in Subsection 2.4.3.

2.6 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 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

Table 2.4: Running times of our algorithms compared with two algorithms of BONMIN (when using formulation (2.9)–(2.13))

		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

try using a "one-tree" approach, such as LP/NLP-based branch-and-bound, instead of OA (see [5, 63]). 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 2.2.2), or in which not all user demands can be met (see Subsection 2.3.2). 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.

Chapter 3

Bi-Perspective Functions for Mixed-Integer Fractional Programs with Indicator Variables

3.1 Introduction

Let y be a vector of n continuous variables, and let $f(y)$ be a real function of y that is defined over a convex domain $C \subseteq \mathbb{R}^n_+$. The *perspective function* of f takes the form $tf(y/t)$, where t is a new continuous variable, and is defined over the domain $t \in \mathbb{R}_+$, $y \in tC$ [30, 66]. (By convention, the perspective function takes the value zero when $t = 0$ and y is the origin.) It is known that the perspective function of $f(y)$ is convex (or concave) if and only if $f(y)$ is convex (or concave) [66].

Perspective functions have a wide range of uses in, e.g., convex analysis, optimisation, statistics, signal processing and machine learning (see [11] for a recent survey). In this paper, we focus on two uses in optimisation. The first is to convert *fractional programs* into convex optimisation problems, and thereby render them easier to solve [10,69]. The second is to reformulate certain *mixed-integer nonlinear programs* (MINLPs), in such a way that the continuous relaxation is strengthened [22, 26]. The MINLPs in question are those with so-called *indicator variables*.

Recently, while studying certain problems arising in the context of mobile wireless communications, we encountered an MINLP that exhibited both of these features (i.e., a fractional objective and indicator variables) simultaneously. It turns out that, in order to obtain tight convex relaxations of such problems, one needs to study a new kind of function, which we call a *bi-perspective* (Bi-P) function. These functions are the topic of this paper.

Unfortunately, it turns out that the Bi-P function of a concave function is not

concave in general. To deal with this, we characterise the concave envelope of a Bi-P function over a rectangular domain. We then derive a family of linear inequalities, which we call *Bi-P cuts*, that completely describe the concave envelope. We also show how to generalise the Bi-P cuts when there are "multiple-choice" constraints, stating that two or more indicator variables cannot take the value 1 simultaneously. Finally, we report the results of some computational experiments. It turns out that, for our particular problem, the new cuts typically close over 95% of the integrality gap.

The paper is structured as follows. The relevant literature is reviewed in Section 3.2. In Section 3.3, we present the theoretical results concerning Bi-P functions and Bi-P cuts. In Section 3.4, we consider the multiple-choice case. In Section 3.5, we give details of our practical application. Computational results are given in Section 3.6, and concluding remarks are made in Section 3.7.

3.2 Literature Review

Now we briefly review the relevant literature. We cover perspective functions and perspective cuts in Subsections 3.2.1 and 3.2.2, respectively. In Subsection 3.2.3, we give some background on optimisation in mobile wireless communications.

3.2.1 Perspective functions

As mentioned in the introduction, two particular uses of perspective functions will turn out to be of importance to us. These are as follows.

1. Consider a *fractional program* of the form:

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

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 the domain C. It is known [10, 69] that such a problem can be reformulated as

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

where t is a new non-negative variable representing $1/g(y)$, and y' is a new vector of variables representing $y/g(y)$. The reformulated problem can often be solved efficiently, since the objective function is concave and the feasible region is convex.

2. Consider an MINLP of minimisation type, in which the cost function contains a term $f(y)$, where y is a vector of continuous variables and f is a convex function. Suppose that the MINLP also contains an *indicator variable*, i.e., 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 MINLP is strengthened, while maintaining convexity, if we replace $f(y)$ with the perspective function $xf(y/x)$ [22, 26]. (For a generalisation, see [7].)

3.2.2 Perspective cuts

One problem with perspective functions is that they are non-differentiable at the origin. Moreover, they become increasingly ill-conditioned as t approaches zero from above. This can cause algorithms for convex optimisation (or indeed convex MINLP) to run into serious numerical difficulties. To get around this, Frangioni & Gentile [22] proposed to approximate perspective functions using linear inequalities. They show that imposing a non-linear constraint of the form $z \geq xf(y/x)$, where f is a convex function and x is an indicator variable, is equivalent to imposing the linear constraints

$$z \geq \nabla f(\bar{y}) \cdot y + \Big(f(\bar{y}) - \nabla f(\bar{y}) \cdot \bar{y} \Big) x \tag{3.1}$$

for all $\bar{y}$ in the domain of y. The constraints (3.1) are called *perspective cuts*. Although the perspective cuts are infinite in number, they can be very useful as cutting planes within an exact algorithm for convex MINLPs with indicator variables (see again [22, 26]).

Note that the classical *Kelley cuts* [38] for the function $f(y)$ take the form

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

Thus, the perspective cuts can be viewed as strengthened Kelley cuts.

3.2.3 Optimisation in mobile wireless communications

In *mobile wireless communications*, mobile devices (such as smartphones or tablets) communicate with one another via so-called *base stations*. Each base station must periodically allocate its available resources (time, power and bandwidth) in order to receive and transmit data efficiently (see, e.g., [17, 28, 64]).

These days, many base stations follow an *Orthogonal Frequency-Division Multiple Access* (OFDMA) architecture. In an OFDMA system, we have a set I of communication channels, called *subcarriers*, and a set J of *users*. Each subcarrier can be

assigned to at most one user, but a user may be assigned to more than one subcarrier. If we let p_i denote the power (in watts) assigned to subcarrier i, the classical *Shannon–Hartley theorem* [72] states that the maximum data rate (in bits per second) that can be transmitted from subcarrier i is:

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

where B_i is the bandwidth of subcarrier i (in hertz), and N_i is the level of noise in channel i (in watts). We remark that $f_i(p_i)$ is concave over the domain $p_i \geq 0$. Moreover, if we let $S_j \subset I$ denote the set of subcarriers allocated to user j, the total data rate for user j is just $\sum_{i \in S_j} f_i(p_i)$.

A wide variety of optimisation problems concerned with OFDMA systems have been considered, with various objectives and side-constraints (see, e.g., [41, 49, 52, 53, 70, 79, 84, 85, 87, 88]). Recently, driven by environmental considerations, some authors working on OFDMA systems have focused on maximising *energy efficiency*, which is defined as total data rate divided by total power (e.g., [79, 84, 85, 88]). This leads immediately to a fractional objective function, which is what led us to the present paper.

3.3 Bi-Perspective Functions and Cuts

This section is concerned with bi-perspective (Bi-P) functions and cuts. In Subsection 3.3.1, we define Bi-P functions and point out that they are neither convex nor concave in general. In Subsection 3.3.2, we show how to compute concave over-estimators of Bi-P functions. Then, in Subsection 3.3.3, we present the Bi-P cuts.

3.3.1 Bi-P functions

To begin, we give a formal definition of Bi-P functions.

Definition 6 *Let y be a vector of n continuous variables, let $f(y)$ be a real function of y that is defined over a convex domain $C \subseteq \mathbb{R}^n_+$, and let t_1 and t_2 be non-negative continuous variables. The function*

$$t_1\, t_2\, f\left(\frac{y}{t_1\, t_2}\right),$$

with domain $t_1, t_2 \in \mathbb{R}_+$, $y \in t_1\, t_2\, C$, will be called the "bi-perspective" (Bi-P) function of $f(y)$. (By convention, the Bi-P function takes the value zero when $t_1\, t_2 = 0$ and y is the origin.)

Whereas standard perspective functions preserve convexity and/or concavity, the same is not true for Bi-P functions.

Lemma 1 *Even if $n = 1$, $f(y)$ is linear and $C = \mathbb{R}_+$, the Bi-P function of $f(y)$ may be neither convex nor concave over its domain.*

Proof. Just let $f(y) = 1$ for all $y \in \mathbb{R}_+$. The Bi-P function is $t_1\, t_2$. Since it is an indefinite quadratic function, it is neither convex nor concave over the domain $y,\, t_1, t_2 \in \mathbb{R}_+$ (or indeed over any convex domain with non-empty interior). □

From now on, we let $f^B(y,\, t_1, t_2)$ denote the Bi-P function of $f(y)$.

3.3.2 Concave envelope

Now suppose that $f(y)$ is concave over the convex domain C. Since the Bi-P function $f^B(y,\, t_1, t_2)$ is not guaranteed to be concave, it is natural to seek the strongest possible concave over-estimating function (sometimes called the concave envelope). We will do this for the case in which the domain of (t_1, t_2) is a square. We will need the following lemma.

Lemma 2 *Let y, $f(y)$, t_1, t_2 and $f^B(y,\, t_1, t_2)$ be defined as in the previous subsection. Let (y^*, t_1^*, t_2^*) lie in the domain of f^B, and let $z^* = f^B\,(y^*,\, t_1^*,\, t_2^*)$. Then, for any constant $\lambda \in \mathbb{R}_+$,*

$$\lambda z^* = f^B\,(\lambda y^*,\, \lambda t_1^*, t_2^*)\,.$$

Proof. We have

$$\begin{aligned}
\lambda z^* &= \lambda\, f^B\,(y^*,\, t_1^*,\, t_2^*) \\
&= \lambda\, t_1^*\, t_2^*\, f\left(\frac{y^*}{t_1^* t_2^*}\right) \\
&= (\lambda\, t_1^*)\; t_2^*\, f\left(\frac{\lambda y^*}{\lambda t_1^*\, t_2^*}\right) \\
&= f^B\,(\lambda y^*,\, \lambda t_1^*, t_2^*)\,.
\end{aligned}$$

□

Theorem 1 *Let y, $f(y)$, C, t_1, t_2 and $f^B(y,\, t_1, t_2)$ be as above, and suppose that $f(y)$ is concave over C. Also suppose that the domain of t_1 is $[a_1, b_1]$, where $0 \le a_1 < b_1$,*

and the domain of t_2 *is* $[a_2, b_2]$*, where* $0 \leq a_2 < b_2$*. Define the following two auxiliary functions:*

$$\begin{aligned} g^1(t_1, t_2) &= a_2 t_1 + b_1(t_2 - a_2) \\ g^2(t_1, t_2) &= a_1 t_2 + b_2(t_1 - a_1). \end{aligned}$$

Then the concave envelope of the Bi-P function of $f(y)$*, over the domain* $t_1 \in [a_1, b_1]$*,* $t_2 \in [a_2, b_2]$ *and* $y \in t_1 t_2 C$*, is:*

$$\min \left\{ g^1(t_1, t_2)\, f\left(\frac{y}{g^1(t_1, t_2)}\right),\ g^2(t_1, t_2)\, f\left(\frac{y}{g^2(t_1, t_2)}\right) \right\}. \tag{3.2}$$

Proof. Let $f^B(y,\, t_1, t_2)$ denote the Bi-P function, and consider the hypograph of the Bi-P function, i.e., the set of all quadruples (y, t_1, t_2, z) such that $t \in [a_1, b_1]$, $t_2 \in [a_2, b_2]$, $y \in t_1\, t_2\, C$ and $z \leq f^B(y, t_1, t_2)$. Lemma 2 implies that, if the point (y^*, t_1^*, t_2^*, z^*) lies on the boundary of the hypograph, then so does the point $(\lambda y^*, \lambda t_1^*, t_2^*, \lambda z^*)$ for all $\lambda \in \mathbb{R}_+$ such that $\lambda t_1^* \in [a_1, b_1]$. In other words, the boundary contain a line segment that connects a point with $t_1 = a_1$ and a point with $t_1 = b_1$. This implies that all extreme points of the boundary satisfy $t_1 \in \{a_1, b_1\}$. A similar argument shows all extreme points also satisfy $t_2 \in \{a_2, b_2\}$. Thus, the extreme points are of four types:

- Type A: $t_1 = a_1$, $t_2 = a_2$;
- Type B: $t_1 = a_1$, $t_2 = b_2$;
- Type C: $t_1 = b_1$, $t_2 = a_2$;
- Type D: $t_1 = b_1$, $t_2 = b_2$.

One can check that:

- $g^1(t_1, t_2)$ is equal to $t_1 t_2$ at all extreme points of type A, C and D, but larger than $t_1 t_2$ at all extreme points of type B;
- $g^2(t_1, t_2)$ is equal to $t_1 t_2$ at all extreme points of type A, B and D, but larger than $t_1 t_2$ at all extreme points of type C.

This implies that:

- The function $g^1\big(t_1, t_2\big)\, f\left(\frac{y}{g^1(t_1, t_2)}\right)$ is equal to $f^B\big(y,\, t_1,\, t_2\big)$ at all extreme points of type A, C and D, and larger than $f^B\big(y,\, t_1,\, t_2\big)$ at all extreme points of type B.

- The function $g^2\big(t_1, t_2\big)\, f\left(\frac{y}{g^2(t_1,t_2)}\right)$ is equal to $f^B\big(y,\, t_1, t_2\big)$ at all extreme points of type A, B and D, and larger than $f^B\big(y,\, t_1, t_2\big)$ at all extreme points of type C.

Thus, the desired concave envelope must be the minimum of those two functions. □

Remark 1 *Theorem 1 is a generalisation of the classical result of McCormick [56], which (in our notation) states that the concave envelope of the quadratic function $t_1\, t_2$ over the domain $t_1 \in [a_1, b_1]$, $t_2 \in [a_2, b_2]$ is given by the minimum of $g^1\big(t_1, t_2\big)$ and $g^2\big(t_1, t_2\big)$.*

Remark 2 *When t_2 is an indicator variable, we have $a_2 = 0$ and $b_2 = 1$. Thus, in this case, $g^1\big(t_1, t_2\big)$ and $g^2\big(t_1, t_2\big)$ reduce to $b_1 t_2$ and $t_1 - a_1\big(1 - t_2\big)$, respectively.*

3.3.3 Bi-P cuts

Observe that the nonlinear function (3.2) is non-differentiable not only when $t_1 t_2 = 0$, but also when $g^1\big(t_1, t_2\big) = g^2\big(t_1, t_2\big)$, i.e., when $\big(t_1, t_2\big)$ is a convex combination of $\big(a_1, a_2\big)$ and $\big(b_1, b_2\big)$. This suggests that standard NLP solvers could struggle to handle functions of the form (3.2). Moreover, there are situations in which one might prefer to use an LP (or MILP) solver rather than an NLP (or MINLP) solver. So, following Frangioni & Gentili [22], we describe the concave envelope by linear inequalities. This is explained in the following proposition.

Proposition 1 *The hypograph of the function (3.2) is described by the linear inequalities*

$$z \;\leq\; \nabla f(\bar{y})\, y \;+\; \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, g^k\big(t_1, t_2\big) \tag{3.3}$$

for $\bar{y} \in C$ and $k = 1, 2$, where the functions g^k are as defined in Theorem 1.

Proof. By definition, the hypograph is the set of quadruples $\big(y, t_1, t_2, z\big)$ satisfying

$$\begin{aligned} z \leq g^k(t_1, t_2)\, f\left(\tfrac{y}{g^k(t_1,t_2)}\right) \qquad & (k = 1, 2) \\ t_k \in [a_k, b_k] \qquad & (k = 1, 2) \qquad (3.4) \\ y \in t_1\, t_2\, C. \qquad & (3.5) \end{aligned}$$

Now, let S denote the set of 6-tuples $\big(y, t_1, t_2, z, t'_1, t'_2\big)$ satisfying (3.4), (3.5), and

$$\begin{aligned} t'_k = g^k(t_1, t_2) \qquad & (k = 1, 2) \qquad (3.6) \\ z \leq t'_k\, f\left(\tfrac{y}{t'_k}\right) \qquad & (k = 1, 2). \end{aligned}$$

Given that the functions g^k are linear, S is just an affine image of the hypograph. Moreover, using exactly the same argument as in Frangioni & Gentile [22], one can show that S is described by the trivial linear constraints (3.4) and (3.6), together with following perspective cuts:

$$z \;\leq\; \nabla f(\bar{y})\, y \;+\; \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, t'_k \qquad (k = 1, 2,\; \bar{y} \in t_1 t_2 C).$$

Eliminating the t'_k variables yields the result. □

We call the inequalities (3.3) *Bi-P cuts*. Note that, when $k = 1$, the Bi-P cuts pass through all points satisfying $a_1 \leq t_1 \leq b_1$, $t_2 = a_2$, $y = t_1 a_2\, \bar{y}$ and $z = t_1 a_2\, f\big(\frac{\bar{y}}{t_1 a_2}\big)$. They also pass through the point $t_1 = b_1$, $t_2 = b_2$, $y = b_1 b_2\, \bar{y}$ and $z = b_1 b_2\, f\big(\frac{\bar{y}}{b_1 b_2}\big)$. Thus, they define maximal faces of the hypograph of the concave envelope whenever the point $\big(\bar{y}, f(\bar{y})\big)$ lies on a maximal face of the hypograph of the original function f. A similar argument applies when $k = 2$.

We now make some remarks about Bi-P cuts.

Remark 3 *If we reduce the domains of t_1 and/or t_2, then the functions g^1 and/or g^2 decrease in value. This in turn causes the Bi-P cuts to become stronger. Thus, if one wishes to make the cuts as tight as possible, it may be worthwhile applying "domain reduction" techniques (see, e.g., [67]) to t_1 and/or t_2.*

Remark 4 *Let $t_1^* \in [a_1, b_1]$, $t_2^* \in [a_2, b_2]$, $y^* \in C$ and $z^* \in \mathbb{R}$ be given. To solve the separation problem for Bi-P cuts, it suffices to compute the quantity (3.2) for the given t_1^*, t_2^* and y^*. If this quantity is less than z^*, then a Bi-P cut is violated. The vector $\bar{y}$ yielding the cut is either $y^*/g^1(t_1^*, t_2^*)$ or $y^*/g^2(t_1^*, t_2^*)$, according to which term in (3.2) is the minimum.*

Remark 5 *When t_2 is an indicator variable, the Bi-P cuts reduce to:*

$$z \;\leq\; \nabla f(\bar{y})\, y \;+\; \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, b_1 t_2 \tag{3.7}$$

$$z \;\leq\; \nabla f(\bar{y})\, y \;+\; \Big(f(\bar{y}) - \nabla f(\bar{y})\, \bar{y}\Big)\, \Big(t_1 - a_1(1 - t_2)\Big). \tag{3.8}$$

We will call constraints (3.7) and (3.8) *type-1* and *type-2* Bi-P cuts, respectively. Note that the type-1 cuts can be derived as standard perspective cuts from the modified perspective function $b_1 t_2 f(y/(b_1 t_2))$. The type-2 cuts, on the other hand, are harder to interpret.

3.4 Bi-P cuts and Multiple-Choice Constraints

It is very common in integer programming to encounter constraints which state that at most one of a set of binary variables can take a positive value. Such constraints go by various names, such as *multiple-choice* constraints, *clique* constraints or *generalised upper bounds*. In this subsection, we consider the case in which each of the variables in the given set is an indicator variable.

To be more precise, suppose we have:

- positive constants a, b with $a < b$;
- an integer $m \geq 2$ and positive integers $n_1, \ldots, n_m$;
- a convex domain $C_j \subseteq \mathbb{R}^{n_j}$ for $j = 1, \ldots, m$;
- a concave function $f_j : \mathbb{R}^{n_j} \to \mathbb{R}$ for $j = 1, \ldots, m$.

Let Q denote the set of quadruples (x, y, z, t) satisfying

$$
\begin{aligned}
t &\in [a, b] && && (3.9)\\
\textstyle\sum_{j=1}^{m} x_j &\leq 1 && && (3.10)\\
z_j &\leq t f_j(y^j/t) && (j = 1, \ldots, m) && (3.11)\\
x_j &\in \{0, 1\} && (j = 1, \ldots, m) && (3.12)\\
y^j &\in t\, C_j && (j = 1, \ldots, m) && (3.13)\\
z_j &\in \mathbb{R} && (j = 1, \ldots, m) && (3.14)\\
x_j = 0 &\Rightarrow z_j = 0 && (j = 1, \ldots, m) &&\\
x_j = 0 &\Rightarrow y^j_i = 0 && (j = 1, \ldots, m;\ i = 1, \ldots, n_j). &&
\end{aligned}
$$

Note that all points in Q satisfy the (non-convex) constraints

$$
\begin{aligned}
y^j &\in t\, x_j\, C_j && (j = 1, \ldots, m)\\
z_j &\leq t x_j f_j\left(\tfrac{y^j}{t x_j}\right) && (j = 1, \ldots, m),
\end{aligned}
$$

provided that, as usual, we use the convention that the right-hand side evaluates to zero when tx_j is zero. Together with Remark 2 in Subsection 3.3.2, this implies that all points in Q satisfy the convex constraints:

$$
\begin{aligned}
z_j &\leq b x_j\, f_j\left(\tfrac{y^j}{b x_j}\right) && (j = 1, \ldots, m) && (3.15)\\
z_j &\leq \big(t - a(1 - x_j)\big)\, f_j\left(\tfrac{y^j}{t - a(1 - x_j)}\right) && (j = 1, \ldots, m). && (3.16)
\end{aligned}
$$

From this, one can derive one family of type-1 and type-2 Bi-P cuts for each value of j.

Perhaps surprisingly, the addition of the constraints (3.15), (3.16) to the system (3.9)–(3.14) does not yield a complete description of the convex hull of Q. This is shown in the following example.

Example3: Let ϵ be a small positive constant. Suppose that (i) $a = \epsilon$ and $b = 1$; and (ii) $n_j = 1$, $C_j = [0, 1]$ and $f_j\left(y_1^j\right) = \left(y_1^j\right)^\epsilon$ for all j. Consider the fractional point obtained by setting t to ϵ, all x and y variables to $1/m$, and all z variables to $\epsilon/(m\epsilon)^\epsilon$. One can check that this point satisfies (3.15) and (3.16) for all j, and therefore all Bi-P cuts. Now, observe that all points in Q satisfy the following convex inequality:

$$\sum_{j=1}^{m} z_j \;\leq\; t\left(\frac{\sum_{j=1}^{m} y_1^j}{t}\right)^\epsilon.$$

The above-mentioned fractional point does not satisfy this, since the left- and right-hand sides evaluate to $(m\epsilon)^{1-\epsilon}$ and $\epsilon^{1-\epsilon}$, respectively. Accordingly, the point cannot lie in the convex hull of Q. □

Now, for any $j \in \{1, \ldots, m\}$ and any $\bar{y}^j \in C^j$, let $||\bar{y}^j||$ denote $f_j(\bar{y}^j) - \nabla f_j(\bar{y}^j) \cdot \bar{y}^j$. The following theorem presents a huge family of valid inequalities, which generalise the type-2 Bi-P cuts (3.8).

Theorem 2 *Let S be any non-empty subset of $\{1, \ldots, m\}$. For each $j \in S$, let $\bar{y}^j$ be any point in C^j such that $||\bar{y}^j|| > 0$. Then the linear inequality*

$$\sum_{j \in S} \frac{z_j}{||\bar{y}^j||} \;\leq\; \sum_{j \in S} \frac{\nabla f_j(\bar{y}^j) \cdot y^j}{||\bar{y}^j||} + t - a\left(1 - \sum_{j \in S} x_j\right) \tag{3.17}$$

is satisfied by all points in Q.

Proof. We consider two cases:

Case 1: $x_j = 0$ for all $j \in S$. This forces y_i^j to be zero for all $j \in S$ and for $i = 1, \ldots, n_j$. This in turn forces z_j to be zero for all $j \in S$. Thus, the inequality reduces to $t \geq a$, which is trivially valid.

Case 2: $x_j = 1$ for some $j \in S$. This forces x_k to be zero for all $k \in S \setminus \{j\}$, along with the associated y and z variables. Thus, the inequality reduces to

$$\frac{z_j}{||\bar{y}^j||} \;\leq\; \frac{\nabla f_j(\bar{y}^j) \cdot y^j}{||\bar{y}^j||} + t.$$

Multiplying this by $||\bar{y}^j||$ we obtain:

$$z_j \;\leq\; \nabla f_j(\bar{y}^j) \cdot y^j \;+ ||\bar{y}^j||\, t.$$

This last inequality can be derived from the (convex) constraint $z_j \leq t f_j(y^j/t)$, in exactly the same way as the perspective cuts. □

We call the constraints (3.17) *multiple-choice* Bi-P cuts, or *MC* cuts for short. They reduce to type-2 Bi-P cuts when $|S| = 1$.

It can be shown that each MC cut defines a face of maximal dimension of the convex hull of Q. It can also be shown that the separation problem for the MC cuts can be solved in polynomial time (assuming that the functions f_j and their partial derivatives can be computed in polynomial time). We omit details, for brevity.

3.5 Application to OFDMA Systems

We now apply the theoretical results in the last section to an optimisation problem associated with OFDMA systems. In Subsection 3.5.1, we define our problem formally and model it as a mixed 0-1 fractional program with indicator variables. In Subsection 3.5.2, we use standard perspective cuts to reformulate the problem as a semi-infinite mixed 0-1 linear program. In Subsection 3.5.3, we show how to strengthen the semi-infinite formulation using Bi-P cuts.

3.5.1 The problem

Let I, J, B_i, N_i and f_i be defined as in Subsection 3.2.3. Let $P > 0$ denote the maximum power available, and let $\sigma \in (0, P)$ denote the *system power*, which is the amount of power needed by the OFDMA system regardless of actual data rates. Finally, suppose that each user $j \in J$ has a non-negative *demand* d_j. The task is to maximise the energy efficiency, subject to the constraint that the total data rate for each user j is at least d_j.

We call this problem the *fractional subcarrier and power allocation problem with rate constraints* (F-SPARC). (A related problem, called the SPARC, was studied in [49]. The difference is that the objective in the SPARC was simply to maximise the total data rate.)

A natural formulation of the F-SPARC is obtained as follows. For all $i \in I$ and $j \in J$, let x_{ij} be a binary variable, indicating whether user j is assigned to subcarrier

i. Also let p_{ij} be a non-negative continuous variable, which represents the amount of power supplied to subcarrier i (if $x_{ij} = 1$), or zero otherwise. We then have:

$$\begin{aligned}
\max \quad & \frac{\sum_{i\in I}\sum_{j\in J} f_i(p_{ij})}{\sigma+\sum_{i\in I}\sum_{j\in J} p_{ij}} && && (3.18)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P-\sigma && && (3.19)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 && (\forall i \in I) && (3.20)\\
& \textstyle\sum_{i\in I} f_i(p_{ij}) \ge d_j && (\forall j \in J) && (3.21)\\
& p_{ij} \le P x_{ij} && (\forall i \in I,\, j \in J) && (3.22)\\
& p_{ij} \in \mathbb{R}_+ && (\forall i \in I,\, j \in J) && (3.23)\\
& x_{ij} \in \{0,1\} && (\forall i \in I,\, j \in J). && (3.24)
\end{aligned}$$

The objective function (3.18) represents the total data rate divided by the total power used (including system power). The constraint (3.19) ensures that the total power used does not exceed the amount available. Constraints (3.20) ensure that each subcarrier is assigned to at most one user. Constraints (3.21) ensure that the user demands are met. Constraints (3.22)–(3.24) are self-explanatory.

The problem (3.18)–(3.24) is a *mixed 0-1 fractional program*. The x variables are clearly *indicator variables*, since setting any x variable to zero forces the corresponding p variable to zero. Moreover, the constraints (3.20) are clearly *multiple-choice* constraints.

From now on, we let $D = \sum_{j\in J} d_j$ denote the total user demand. Note that an upper bound for the F-SPARC can be computed by solving the following (continuous) fractional program:

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

This fractional program can be converted into a convex program using the transformation mentioned in Subsection 3.2.1. We denote the corresponding upper bound by U.

3.5.2 Reformulation

We now reformulate the problem to make it easier to solve. This is done in three steps.

The first step is to convert the fractional objective function into a concave function. To do this, we use the transformation mentioned in Subsection 3.2.1. Let t be a

non-negative continuous variable, representing $1/(\sigma + \sum_{i\in I}\sum_{j\in J} p_{ij})$, i.e., the reciprocal of the total power used. Also, for all i and j, let $\tilde{p}_{ij}$ be a non-negative continuous variable, representing $t\,p_{ij}$. The problem is then equivalent to the following:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} t f_i(\tilde{p}_{ij}/t) && && (3.25)\\
\text{s.t.} \quad & (3.20), (3.24) && && \\
& \sigma t + \textstyle\sum_{i\in I}\sum_{j\in J} \tilde{p}_{ij} = 1 && && (3.26)\\
& 1/P \;\le\; t \;\le\; 1/\sigma && && (3.27)\\
& \textstyle\sum_{i\in I} f_i(\tilde{p}_{ij}/t) \ge d_j && (\forall j \in J) && (3.28)\\
& 0 \;\le\; \tilde{p}_{ij} \;\le\; x_{ij} && (\forall i \in I,\, j \in J). && (3.29)
\end{aligned}$$

The objective function (3.25) is now concave, and all constraints are linear, apart from (3.28), which are convex.

The second step is the following. For $i \in I$ and $j \in J$, define a new variable, say z_{ij}, representing the quantity $t f_i(\tilde{p}_{ij}/t)$. The problem then becomes:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} z_{ij} && && \\
\text{s.t.} \quad & (3.20), (3.24), (3.26), (3.27), (3.29) && && \\
& \textstyle\sum_{i\in I} z_{ij} \ge d_j t && (\forall j \in J) && \\
& z_{ij} \;\le\; t f_i(\tilde{p}_{ij}/t) && (\forall i \in I, j \in J) && (3.30)\\
& z_{ij} \in \mathbb{R}_+ && (\forall i \in I, j \in J). &&
\end{aligned}$$

Now, all of the nonlinearity has been "concentrated" in the (convex) constraints (3.30).

Finally, we replace the constraints (3.30) with the following linear constraints:

$$z_{ij} \;\le\; f_i'(\bar{p})\,\tilde{p}_{ij} + (f_i(\bar{p}) - f_i'(\bar{p})\,\bar{p})\,t \qquad (\forall i \in I,\, j \in J,\, \bar{p} \in [0, P-\sigma]). \tag{3.31}$$

These linear constraints can be derived in the same way as standard perspective cuts. Since they are infinite in number, we have formulated the F-SPARC as a *semi-infinite mixed 0-1 linear program*. It can be solved (to arbitrary fixed accuracy) with an LP-based branch-and-cut algorithm.

3.5.3 Bi-P cuts

We now use the results in Section 3.3 to strengthen our semi-infinite formulation of the F-SPARC.

We start by reducing the domain of t. Recall that D denotes the total user demand. We compute the following lower bound on the total amount of power used:

$$P_{\min} \;=\; \sigma \;+\; \min\left\{ \sum_{i\in I} p_i : \; \sum_{i\in I} f_i(p_i) \geq D, \; p_i \geq 0 \; (i \in I) \right\}.$$

The upper bound on t in (3.27) can then be reduced from $1/\sigma$ to $1/P_{\min}$.

Now, let i and j be fixed, and consider the constraints (3.24), (3.29) and (3.30). It is clear that all feasible triples $(x_{ij}, \tilde{p}_{ij}, r_{ij})$ satisfy

$$z_{ij} \;\leq\; t\, x_{ij} f_i\left(\frac{\tilde{p}_{ij}}{t x_{ij}}\right),$$

where, as usual, the convention is that the function on the right-hand side takes the value zero when $\tilde{p}_{ij} = t x_{ij} = 0$. The function is clearly a Bi-P function, in which t plays the role of t_1 and x_{ij} plays the role of t_2. The only tricky part is that the natural domain of $\tilde{p}_{ij}$ is $[0, 1]$, which is not equal to $t\, x_{ij}\, C$ for some convex domain C. Despite this complication, we have the following result:

Proposition 2 *For all $i \in I$, $j \in J$ and $\bar{p} \in [0, P - \sigma]$, the following type-1 and type-2 Bi-P cuts are valid for the F-SPARC:*

$$z_{ij} \;\leq\; f_i'(\bar{p})\, \tilde{p}_{ij} + \left(\frac{f_i(\bar{p}) \;-\; f_i'(\bar{p})\, \bar{p}}{P_{\min}}\right) x_{ij} \tag{3.32}$$

$$z_{ij} \;\leq\; f_i'(\bar{p})\, \tilde{p}_{ij} + (f_i(\bar{p}) - f_i'(\bar{p})\, \bar{p}) \left(t - \frac{1 - x_{ij}}{P}\right). \tag{3.33}$$

Proof. We already know that t must lie in the interval $[1/P, \, 1/P_{\min}]$. Also, $p_{ij} \leq P - \sigma$, or, equivalently, $\tilde{p}_{ij} \in t\,[0, P - \sigma]$. In fact, given that $\tilde{p}_{ij}$ must be zero when x_{ij} is zero, we can conclude that $\tilde{p}_{ij} \in t\, x_{ij}\, [0, P - \sigma]$. We can now apply Remark 5. □

Note that the type-2 cuts (3.33) dominate (3.31).

Finally, since the constraints (3.20) are multiple-choice constraints, we can also generate MC cuts. In our preliminary experiments, we found that the most useful MC cuts, by far, were those with (a) $S = J$ and (b) $\bar{p}_{ij}$ equal to the same (positive) value for all $j \in J$. These cuts take the form:

$$\sum_{j\in J} z_{ij} \;\leq\; f_i'(\bar{p}) \sum_{j\in J} \tilde{p}_{ij} \;+ \Big(f_i(\bar{p}) - f_i'(\bar{p})\bar{p}\Big)\, t \quad \big(i \in I, \, \bar{p} \in [0, P - \sigma]\big). \tag{3.34}$$

A possible explanation for the effectiveness of the cuts (3.34) is that they collectively enforce the following convex inequalities:

$$\sum_{j \in J} z_{ij} \leq t \, f_i \left(\frac{\sum_{j \in J} \tilde{p}_{ij}}{t} \right) \qquad (i \in I). \tag{3.35}$$

From this it can be shown that the upper bound obtained by adding all of the cuts (3.34) to the continuous relaxation of the semi-infinite formulation is equal to U, the upper bound mentioned at the end of Subsection 3.5.1. In practice, this bound tends to be quite tight.

We remark that the separation problem for the cuts (3.34) can be solved simply by setting $\bar{p}$ to the current value of $\sum_{j \in J} \tilde{p}_{ij}$, and checking the inequality (3.35) for violation. This can be done in $O(|J|)$ time for a given i.

3.6 Computational Experiments

In the previous section, we described three families of cutting planes for the F-SPARC: the type-1 cuts (3.32), the type-2 cuts (3.33), and the special MC cuts (3.34). In this section, we present some computational results to shed light on the relative usefulness of these different kinds of cuts.

3.6.1 Test instances

To construct our test instances, we used the procedure described in [49], which is designed to produce instances typical of a small base station. In detail, the instances have $|I| = 72$, $|J| \in \{4, 6\}$, and P set to 36 watts. The noise powers N_i are random numbers distributed uniformly in $(10^{-6}, 10^{-5})$, and the bandwidths B_i are all set to 1.25MHz. The user demands follow a lognormal distribution.

Let D be the total demand, as before, and let M be the maximum possible data rate of the system. As in [49], we call the quantity D/M the *demand ratio* (DR) of the given instance. The user demands are scaled so that the DR takes values in $\{0.75, 0.8, 0.85, 0.9, 0.95\}$. (The closer the DR is to 1, the harder the instance tends to be.) For each combination of $|J|$ and DR, we constructed 10 random instances. This makes $2 \times 5 \times 10 = 100$ instances in total. These instances have been made available at:

`http://www.research.lancs.ac.uk/portal/en/datasets/search.html`

under "OFDMA Optimisation".

3.6.2 Experimental setup

For each instance, we did the following. We began by computing the upper bound U mentioned at the end of Subsection 3.5.1, using `MOSEK`. We then ran the heuristic described in [50] to compute a lower bound. If the bounds differed by more than 0.1%, we ran an exact algorithm, similar to the one described in [49], until the difference between the lower and upper bounds dropped below 0.1%. We remark that this exact algorithm took a long time (sometimes several minutes) to converge for some of our test instances.

The next step was to solve, for each instance, the continuous relaxation of the formulation described in Subsection 3.5.2. To do this, we used a (fairly standard) LP-based cutting-plane algorithm, in which the inequalities (3.31) were added dynamically as cutting planes. In each major iteration, all inequalities violated by more than 10^{-4} were added to the LP. A time limit of two minutes per instance was also imposed. (In most cases, tailing off occurred well before the time limit.)

The cutting-plane 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 used the LP solver from the CPLEX 12.6.3 Callable Library (with default settings). More specifically, we used primal simplex to solve the initial relaxation and dual simplex to re-optimise after adding cuts.

Finally, we ran the cutting-plane algorithm again for each instance, switching on and off various combinations of the cuts (3.32)–(3.34). The purpose of this was to enable us to identify the cuts that tend to be most useful in practice.

3.6.3 Results

We present results for six versions of the cutting-plane algorithm:

- "$\emptyset$": Constraints (3.31) alone, without any Bi-P cuts.
- "T1": Constraints (3.31) plus type-1 Bi-P cuts (3.32).
- "T2": Type-2 Bi-P cuts (3.33) alone.
- 'MC": Constraints (3.31) plus MC cuts (3.34).
- "T1+T2": Type-1 cuts (3.32) and type-2 cuts (3.33).
- "All": Type-1 cuts (3.32), type-2 cuts (3.33) and MC cuts (3.34).

Table 3.1: Average percentage gaps when $|J| = 4$

	Demand Ratio				
Cuts	0.75	0.80	0.85	0.90	0.95
$\emptyset$	165	165	168	172	175
T1	0.00	0.00	0.09	0.03	0.02
T2	0.01	0.01	0.03	0.01	0.03
MC	2.19	1.39	1.06	0.25	0.06
T1, T2	0.00	0.00	0.03	0.01	0.01
All	0.00	0.00	0.02	0.01	0.01

Table 3.2: Average percentage gaps when $|J| = 6$

	Demand Ratio				
Cuts	0.75	0.80	0.85	0.90	0.95
$\emptyset$	275	276	279	286	291
T1	0.08	0.08	0.14	0.15	0.05
T2	0.01	0.01	0.04	1.64	0.81
MC	2.30	2.41	1.63	0.33	0.15
T1, T2	0.01	0.01	0.04	0.15	0.03
All	0.01	0.01	0.04	0.14	0.03

Tables 3.1 and 3.2 show, for each set of 10 instances and each combination of cutting planes, the average gap between the upper bound from the cutting-plane algorithm and the lower bound from the exact algorithm, expressed as a percentage of the lower bound.

From the tables, we see that the unstrengthened cuts (3.31) lead to extremely poor upper bounds in all cases. Moreover, the gaps for $|J| = 4$ are much worse than the gaps for $|J| = 6$. Type-1, type-2 and MC cuts all close the gap considerably, but type-1 and type-2 cuts appear to be more effective than MC cuts. Using type-1 and type-2 cuts in combination is particularly effective. In fact, the benefit gained by including MC cuts as well is rather small.

3.7 Concluding Remarks

Perspective reformulations and cuts are an invaluable tool for both fractional programming and MINLP with indicator variables. We have shown that, when one is dealing with a mixed-integer fractional program with indicator variables, one needs to use "bi-perspective" reformulations and cuts in order to obtain bounds that are

useful within an exact solution algorithm. We believe that extensions of perspective reformulations and cuts to other classes of problems would be a valuable topic for future research.

As for our specific application, to optimisation in OFDMA systems, we plan to look next at *stochastic dynamic* variants of the problem, in which users arrive and depart at random over time.

Chapter 4

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

4.1 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., [17]). 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., [41, 47–49, 52, 53, 70, 79, 84, 85, 87, 88]). Unfortunately, it turns out that most of these problems are $\mathcal{NP}$-hard [32, 52, 53]. Thus, most authors resort to heuristics. In our recent papers [48, 49], however, we presented exact solution algorithms based on mixed-integer nonlinear programming (MINLP). The problem considered in [49] 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 [48] 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 [49] is capable of solving many realistic problem instances to proven optimality (or near-optimality) within a couple of seconds. The algorithm in [48], 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 [48]. 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 4.2, the heuristic is presented in Section 4.3, the computational results are given in Section 4.4, and concluding remarks are made in Section 4.5.

To make the paper self-contained, we recall the following result from [68] (see also [10, 69]). 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

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

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.

4.2 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* [72] 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 [48], 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:

$$\begin{aligned}
\max \quad & \frac{\sum_{i\in I}\sum_{j\in J} r_{ij}}{\sigma+\sum_{i\in I}\sum_{j\in J} p_{ij}} && && (4.1)\\
& \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P-\sigma && && (4.2)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 && (\forall i \in I) && (4.3)\\
& \textstyle\sum_{i\in I} r_{ij} \ge d_j && (\forall j \in J) && (4.4)\\
& r_{ij} \le f_i(p_{ij}) && (\forall j \in J) && (4.5)\\
& p_{ij} \le (P-\sigma)x_{ij} && (\forall i \in I,\, j \in J) && (4.6)\\
& x_{ij} \in \{0,1\} && (\forall i \in I,\, j \in J) &&\\
& p_{ij}, r_{ij} \in \mathbb{R}_+ && (\forall i \in I,\, j \in J). &&
\end{aligned}$$

The objective function (4.1) represents the total data rate divided by the total power (including the system power). The constraint (4.2) enforces the limit on the total power. Constraints (4.3) ensure that each subcarrier is assigned to at most one user. Constraints (4.4) ensure that user demands are met. Constraints (4.5) ensure that the data rate for each subcarrier does not exceed the theoretical limit. Constraints (4.6) 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 (4.1) and the constraints (4.5) are both nonlinear, but they are easily shown to be concave and convex, respectively. The exact algorithm in [48] 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* [63], 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.

4.3 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.

4.3.1 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:

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

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., [12, 28].)

Now consider the following fractional program:

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

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:

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

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:

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

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.

4.3.2 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 2. When Algorithm 2 terminates, if L and U are sufficiently close (say, within 1%), then we have solved the instance (to the desired tolerance).

4.3.3 Improving by Reallocating Power

Algorithm 2 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:

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

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 $1+\epsilon$.

Algorithm 2: Binary search heuristic for F-SPARC

input: power P, bandwidths B_i, noise powers N_i,
demands d_j, system power σ, tolerance $\delta > 0$.

Compute total user demand $D = \sum_{j \in J} d_j$;
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 $L := 0$, $\epsilon_\ell := 0$ and $\epsilon_u := (M/D) - 1$;
repeat
  Set $\epsilon := (\epsilon_\ell + \epsilon_u)/2$;
  Solve `NLP2`(ϵ). Let $(\tilde{p}^*, \tilde{r}^*, t^*)$ be the solution and L' its efficiency;
  Solve `01LP` with r^* set to $\tilde{r}^*/t^*$;
  if `01LP` *is infeasible* **then**
    Set $\epsilon_\ell := \epsilon$;
  else
    Let x^* be the solution to `01LP`;
    if $L' > L$ **then**
      Set $L := L'$, $\bar{p} := \tilde{p}^*/t^*$ and $\bar{x} := x^*$;
    end
    Set $\epsilon_u := \epsilon$;
  end
until $\epsilon_u - \epsilon_\ell \leq \delta$ *or* $L \geq U/(1+\delta)$;
if $L > 0$ **then**
  Output feasible solution $(\bar{x}, \bar{p})$;
else
  Output "No feasible solution was found.";
end

4.4 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.

4.4.1 Test Instances

To construct our test instances, we used the procedure described in [49], 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 [48], 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.

4.4.2 Results

Table 4.1 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

	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 4.1: Number of instances where heuristic failed to find a feasible solution

	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 4.2: Average percentage gap between lower and upper bounds.

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.

Table 4.2 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.)

Finally, Table 4.3 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.)

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

	Demand Ratio					
$\|J\|$	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 4.3: Average time (in seconds) taken by the heuristic when it succeeded.

exact algorithm, we can report that the heuristic is typically faster by at least two orders of magnitude.

4.5 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.

Chapter 5

A Heuristic for Dynamic Resource Allocation in Overloaded OFDMA Systems

5.1 Introduction

In modern mobile wireless communication systems, the base stations often use a coding scheme called *Orthogonal Frequency-Division Multiple Access* or OFDMA (see, e.g., [17]). In OFDMA, there are a number of transmission channels, called *subcarriers*. At any given point in time, there is a set of *users* with known demands. Each subcarrier must be assigned to only one user, but a user may be assigned to more than one subcarrier. The data rate for each subcarrier is a nonlinear function of the power allocated to it, and there is a limited amount of power available.

There are actually many different optimisation problems associated with OFDMA systems, with various objective functions, side-constraints and planning horizons (see, e.g., [16, 41, 42, 47–50, 55, 59, 65, 70, 74–76, 80, 83, 84, 86, 88]). Most of them have been shown to be $\mathcal{NP}$-hard [32, 52, 53].

In our earlier paper [49], we considered a relatively simple problem, in which the set of users is treated as fixed. The problem is to allocate the power to the subcarriers, and the subcarriers to the users, in order to maximise the overall data rate, subject to satisfying the demand of each user. We called this the *joint subcarrier and power allocation problem with rate constraints* (SPARC), and presented an exact algorithm for it.

Now, let M be the maximum data rate achieveable by the system (in megabits per second, Mb/s), and let D be the total demand (again in Mb/s). When $D/M \leq 0.93$, the algorithm in [49] is very fast, taking only a fraction of a second. On the other hand,

when $D/M > 0.93$, the algorithm becomes unacceptably slow, sometimes taking minutes to find a feasible solution (or prove infeasibility).

In this paper, we address the high-demand case. More precisely, let $D(t)$ denote the demand at time t. We are concerned with the situation in which $D(t)/M$ regularly exceeds 0.93. In this case, we say that the system is *overloaded*. It turns out that a very different approach is needed for the overloaded case. This is for several reasons:

1. At certain points in time, the system may not have the capacity to satisfy all of the users.

2. Thus, we may need to be content with only *partially* satisfying the users at certain times.

3. This in turn means that we must ensure that users are treated in a manner that is perceived to be *fair*.

4. Since an exact approach is likely to be too slow, we must use a *heuristic* approach.

5. To be of practical use, the heuristic must be able to *re-optimise quickly*, as users arrive and depart. (Equivalently, it must be suitable for a *stochastic dynamic* optimisation problem rather than a static one.)

To address these considerations, we devise a heuristic, based on the solution of a single small convex program, followed by the periodic application of local search. The neighbourhoods are specially designed so that we can search them within a fraction of a second. Extensive simulations on realistic data indicate that the heuristic is fast enough to be used in real-time, and consistently delivers allocations of very good quality (according to various quality measures).

We remark that our heuristic is most appropriate for so-called *non-delay-constrained* traffic (such as emails and file requests), for which occasional delays are acceptable. For *delay-constrained traffic* (such as phone calls and live video), our heuristic may be less useful. (See Tao *et al.* [76] for more on these two kinds of traffic.)

The paper is structured as follows. Section 5.2 contains a brief literature review. Section 5.3 describes the new problem in detail, and Section 5.4 describes the heuristic itself. The computational results are given in Section 5.5 and some final remarks are made in Section 5.6.

Throughout the paper, we let I denote the set of subcarriers. Each subcarrier $i \in I$ has a known *bandwidth* B_i (measured in MHz) and a known *noise power* N_i (in

watts). The set of users at time t is denoted by $J(t)$. The demand of user $j \in J(t)$, in Mb/s, is denoted by d_j. The total demand at time t is denoted by $D(t)$. That is, $D(t) = \sum_{j \in J(t)} d_j$. When we are considering only a single time period, we drop the index t and just write J and D, respectively. The amount of power available, in watts, is denoted by P. We assume that the demand process is stationary, so that the expected value of $D(t)$ is constant over time. Borrowing from queueing theory parlance, we call the expected value of $D(t)/M$ the *traffic intensity* and denote it by ρ.

5.2 Literature Review

As mentioned in the introduction, there is by now an extensive literature on optimisation in OFDMA systems. For the sake of brevity, we review here only a few works of direct relevance.

Consider a single subcarrier $i \in I$. The classical *Shannon-Hartley theorem* [72] states that, if we allocate p watts of power to subcarrier i, then the maximum possible data rata achievable via subcarrier i, in bits per second, is

$$f_i(p) = B_i \log_2 \left(1 + \frac{p}{N_i}\right).$$

We remark that this function is concave (over the domain $\mathbb{R}_+$).

Now consider the case of multiple subcarriers, and recall that M denotes the maximum data rate achievable by the system. One can compute M quickly by solving the following NLP:

$$\max \left\{ \sum_{i \in I} f_i(p_{ij}) : \sum_{i \in I} p_i \leq P,\, p \in \mathbb{R}_+^{|I|} \right\}. \tag{5.1}$$

This NLP can be solved quickly using a method called *water filling* (see, e.g., [12, 17, 28]).

Now we recall the formulation of the SPARC presented in [49]. This formulation considers only a single time period. For each $i \in I$ and $j \in J$, let x_{ij} be a binary variable, taking the value 1 if and only if subcarrier i is assigned to user j, and 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. The SPARC is then formulated as the

following mixed 0-1 convex program:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{j\in J} f_i(p_{ij}) & & \\
\text{s.t.} \quad & \textstyle\sum_{i\in I}\sum_{j\in J} p_{ij} \le P & & (5.2)\\
& \textstyle\sum_{i\in I} f_i(p_{ij}) \ge d_j & (j\in J) & (5.3)\\
& \textstyle\sum_{j\in J} x_{ij} \le 1 & (i\in I) & (5.4)\\
& p_{ij} \le P x_{ij} & (i\in I,\, j\in J) & (5.5)\\
& p_{ij} \in \mathbb{R}_+ & (i\in I,\, j\in J) & \\
& x_{ij} \in \{0,1\} & (i\in I,\, j\in J). &
\end{aligned}$$

The constraint (5.2) imposes the limit on the total available power. The constraints (5.3) ensure quality of service (QoS). The constraints (5.4) ensure that each subcarrier is allocated to at most one user. The constraints (5.5), which are the *variable upper bounds*, ensure that x_{ij} takes the value 1 if $p_{ij} > 0$. The remaining constraints are self-explanatory.

As mentioned in the introduction, the algorithm in [49] works well when $D/M \le 0.93$, but is slow otherwise. Moreover, when $0.93 < D/M \le 1$, there is a chance that the SPARC is infeasible. We conclude that the approach in [49] is suitable only when (a) the user demands are more-or-less static, and/or (b) $D(t)/M$ rarely exceeds 0.93.

Finally, we mention that there is a stream of literature on *fairness* in multi-user communications systems (see, e.g., [16, 33, 35, 39, 55, 58, 59, 74, 75, 80]). As mentioned above, fairness will be relevant to us because, when the traffic intensity is high, we may not be able to satisfy the demands of all users.

5.3 A Stochastic Dynamic Version of the SPARC

It turns out that some thought is needed before one can formally define a stochastic dynamic version of the SPARC. In particular, one must consider (i) what constitutes an instance of the problem, and (ii) which function is to be optimised. These issues are covered in the following two subsections.

5.3.1 Instance data

As in the standard SPARC, we assume that the set of subcarriers I is fixed, and that we are given the bandwidths B_i, noise powers N_i, and power limit P. (In real-life systems, the N_i may fluctuate a little over time. Our approach can be extended to

cover that case, but we do not give details, for brevity.) As for the users, we make the following assumptions:

- User arrivals are Markovian with known average rate λ (per second).
- The durations of the user requests are i.i.d., with known probability distribution and known mean $\bar{t}$ (in seconds).
- The user demands follow a known probability distribution with known mean $\bar{d}$ (in Mb/s).

One can check that, at steady-state, the expected number of users is $\lambda \bar{t}$ and the expected total user demand is $\bar{D} = \lambda \bar{t} \bar{d}$. For the traffic intensity, we have $\rho = \bar{D}/M$. In our preliminary experiments, we found that ρ is a reasonably reliable measure of the difficulty of an instance. (Other obvious potential drivers of difficulty are the variances of the user durations and user demands, but we did not find these to be so important in our simulations.)

5.3.2 Objective function

Some thought also needs to be paid to the objective function. In particular, one must address the issue of fairness mentioned in the introduction.

Now, let us temporarily consider the static case, in which all user demands are known. Let $p \in \mathbb{R}_+^{|I||J|}$ be a fixed power allocation. For each user $j \in J$, we define the *user rate* $r_j = \sum_{i \in I} f_i(p_{ij})$ and the *satisfaction* $s_j = r_j/d_j$. Then, the demand of a user is met if and only if the satisfaction is at least one. A natural objective is then to maximise the mean of the s_j.

Unfortunately, the use of this "max-mean" objective can lead to very unfair solutions when the user demands have a wide range.

Example: Suppose that $|J| = 2$, and that the demands are 10 and 1. Suppose that $|I| = 3$, and the subcarriers have data rates of 8, 2 and 1, respectively. If we assign subcarriers 1 and 2 to user 1 and subcarrier 3 to user 3, the mean satisfaction will be 1. But if we assign only subcarrier 1 to user 1 and subcarriers 2 and 3 to user 3, the mean satisfaction will be 1.9. So the second vector is preferable according to the "max-mean" criterion, even though the first allocation completely satisfies both users. □

To avoid such unfair solutions, one could attempt to maximise the minimum satisfaction instead. Unfortunately, 'max-min' optimisation problems are notoriously difficult to solve, by either exact or heuristic methods, because the objective function is 'flat' (i.e., small changes in p may lead to no change in the value of the objective).

After some experimentation, we discovered the following alternative objective function:

Definition 7 *The "weighted harmonic mean satisfaction" (WHMS) is:*

$$\frac{D}{\sum_{j\in J} d_j s_j^{-1}} \;=\; \frac{D}{\sum_{j\in J} d_j^2/r_j}.$$

In our experience, maximising the WHMS tends to lead to solutions that perform very well according to the max-min criterion. Indeed, in the above example, the first solution has a WHMS of $11/(10+1) = 1$, whereas the second (unfair) solution has a WHSM of $11/\left(\frac{100}{8} + \frac{1}{3}\right) = 6/7$. The following proposition gives a partial explanation for this phenomenon.

Proposition 3 *Let $d \in \mathbb{R}_+^{|J|}$ be a demand vector and let R be a positive constant. Consider the following two continuous optimisation problems: the "max-min" problem*

$$\max\left\{\min_{j\in J}\{r_j/d_j\} : \sum_{j\in J} r_j = R,\, r_j > 0\,(j\in J)\right\}$$

and the "max WHMS" problem

$$\max\left\{\frac{D}{\sum_{j\in J} d_j^2/r_j} : \sum_{j\in J} r_j = R,\, r_j > 0\,(j\in J)\right\}.$$

These two problems have the same optimal solutions.

Proof. The solution to the max-min problem is to set r_j to $d_j R/D$ for all j. This gives each user a satisfaction of R/D. Now, since D is fixed, the max WHMS problem is equivalent to

$$\min\left\{\sum_{j\in J} d_j^2/r_j : \sum_{j\in J} r_j = R,\, r_j > 0\,(j\in J)\right\}.$$

We solve this last problem using the method of Lagrange multipliers. We give the constraint $\sum_{j\in J} r_j = R$ a Lagrange multiplier λ and consider the Lagrangian

$$L(r,\lambda) \;=\; \sum_{j\in J} d_j^2/r_j \;+\; \lambda\left(\sum_{j\in J} r_j - R\right).$$

We now have

$$\partial L(r,\lambda)/\partial r_j \;=\; \lambda - d_j^2/r_j^2 \qquad\qquad (j \in J).$$

Setting these partial derivatives to zero, we obtain $d_j^2/r_j^2 = \lambda$ for all j, or, equivalently, $r_j = \sqrt{\lambda}/d_j$ for all j. Thus, the optimal r values are proportional to $1/d_j$. In other words, r_j is set to $d_j R/D$ for all j, just as in the max-min solution. □

Now let us return to the stochastic dynamic case. In light of the above, one might wish to compute a policy that maximises the *expected* WHMS, where the expectation is taken over an infinite number of time periods. Unfortunately, this looks like an extremely difficult task, especially in the overloaded case. So, as mentioned in the introduction, we content ourselves with a heuristic approach that updates the resource allocation in each time period.

5.4 The Heuristic

In this section, we present a heuristic for maximising the expected WHMS when the system is overloaded.

5.4.1 Initial solution

Before one can apply local search, one needs an initial solution to start from. To construct an initial solution, we use the greedy heuristic described in Algorithm 3. During the course of the algorithm, r_j is the current data rate given to user j. The choice of the factor of $\sqrt{d_j}$ is designed to make it more likely that channels with high data rate will be allocated to users with high demand, yet still ensure that at least some of the demand of each user is satisfied.

5.4.2 Local search

To improve the initial solution, we use a straighforward local search heuristic. This heuristic consists of two main phases, as described in Algorithms 4 and 5.

In the first phase, we take each subcarrier and check if it should be assign it to another user. This phase can be implemented to run in only $O(|I|\,|J|)$ time. Indeed, maximising the WHMS is equivalent to minimising

$$\sum_{j\in J} \frac{d_j^2}{\sum_{i\in I} f_i(p_{ij})}. \tag{5.6}$$

Algorithm 3: Greedy Constructive Heuristic

Input: bandwidths B_i, noise powers N_i, initial demands d_j.

Solve the NLP (5.1) and let p^* be the optimal solution;
Sort the channels in non-increasing order of $f_i(p_i^*)$ and let L be the sorted list;
for *each user* $j \in J$ **do**
  Set $r_j := 0$;
end
for *each channel* i *in the list* L **do**
  Assign channel i to the user with the smallest value of $r_j/\sqrt{d_j}$;
  (In case of ties, assign it to the user with highest d_j);
  Increase r_j by $f_i(p_i^*)$;
end
Output: Initial allocation of channels to users.

Algorithm 4: First Improvement Phase

Input: bandwidths B_i, noise powers N_i, demands d_j,
fixed power allocation vector p^*,
current subcarrier allocation, current data rates r_j.

for *each subcarrier* $i \in I$ **do**
  Let k be the user to which subcarrier i is currently allocated;
  for *each user* $j \in J \setminus \{k\}$ **do**
    if *the WHMS can be improved by re-allocating* i *to* j **then**
      Re-allocate subcarrier i to user j;
      Update r_k and r_j;
    end
  end
end
Output: Improved subcarrier allocation.

Algorithm 5: Second Improvement Phase

Input: bandwidths B_i, noise powers N_i, demands d_j,
fixed power allocation vector p^*,
current subcarrier allocation, current data rates r_j.

for *each pair of subcarriers* $\{i, i'\} \subset I$ **do**
  Let k, k' be the users to which the subcarriers are currently allocated;
  if $k \neq k'$ *and the WHMS can be improved by swapping the allocation of subcarriers* i *and* i' **then**
    Swap the allocation of subcarriers i and i';
    Update r_k and $r_{k'}$;
  end
end
Output: Improved subcarrier allocation.

If we take channel i and assign it to user j instead of user k, the function (5.6) will increase by

$$\frac{d_k^2}{r_k - p_i^*} - \frac{d_k^2}{r_k} + \frac{d_j^2}{r_j + p_i^*} - \frac{d_j^2}{r_j}.$$

If this is negative, then we can accept the proposed move. We can check this in constant time for a given i and j.

In the second phase, we take pairs of subcarriers and swap the users. This phase can be implemented to run in $O\big(|I|^2\big)$ time. Indeed, if subcarriers i and i' are assigned to users k and k', respectively, and we swap the allocation, the function (5.6) will increase by

$$\frac{d_k^2}{r_k + p_{i'}^* - p_i^*} - \frac{d_k^2}{r_k} - \frac{d_{k'}^2}{r_{k'} + p_i^* - p_{i'}^*} + \frac{d_{k'}^2}{r_{k'}}.$$

5.4.3 Extension to the dynamic case

To adapt our heuristic to the dynamic case, we basically run the local search heuristic periodically. Details are given in Algorithm 6. The key idea is that, if the set of users has changed, we restore feasibility and re-optimise as quickly as possible. In particular, we do not call Algorithms 4 and 5 more than once in any given time period. Then, with appropriate data structures, the time taken by the algorithm in each time period is only $O(|I|^2)$. This limit on the running time is necessary, since, in the real system, one needs to decide how to re-allocate the subcarriers in a fraction of a second.

5.5 Experiments

In this section, we report on some computational experiments that we conducted. The heuristic described in the previous section was coded in Julia v0.5 and run on an intel Core i7 3.1 GHz CPU, with 16GB of RAM, under Ubuntu 16.04.1 LTS. The program calls on MOSEK 7.1 (with default settings) to solve the initial NLP.

5.5.1 Test Instances

We took particular care to make our test instances as realistic as possible, based on the IEEE 802.16 standard. With regard to subcarriers, we set $|I| \in \{72, 180, 300\}$. The noise powers N_i are random numbers distributed uniformly in the open interval $(0, 10^{-10})$, and the bandwidths B_i are all set to 2.5MHz. As for users, we assume that (a) the inter-arrival times follow a negative exponential distribution, (b) the

Algorithm 6: Dynamic Local Search Heuristic

Input: bandwidths B_i, noise powers N_i, initial set of users $J(0)$, initial demands d_j, number of time periods T.

```
Construct an initial solution using Algorithms 3, 4 and 5;
for t = 1, ..., T do
    Let J(t) be the current set of users;
    Let J^- = J(t-1) \ J(t);
    if J^- ≠ ∅ then
        for j ∈ J^- do
            for each subcarrier that was allocated to user j do
                Re-allocate the subcarrier to an arbitrary user in J(t);
            end
        end
    end
    Let J^0 be the set of users in J(t) that currently have no subcarriers
     allocated to them;
    if J^0 ≠ ∅ then
        Let J^+ contain all users in J(t) that currently have two or more
         subcarriers assigned to them;
        for j ∈ J^0 do
            Let j^+ be the user in J^+ with the highest satisfaction;
            Let i be a subcarrier that was allocated to user j^+;
            Re-allocate subcarrier i to user j;
            if user j^+ now has only one subcarrier then
                Remove j^+ from J^+;
            end
        end
    end
    Re-optimise by calling Algorithms 4 and 5;
end
```

service times (in seconds) are uniformly distributed in $[1, 4]$, and (c) the demands d_j are obtained by sampling random numbers from a unit lognormal distribution, and multiplying by a positive constant c. Finally, the power limit P is set to $|I|/2$, in watts.

Note that, if the mean arrival rate (in users per second) is λ, then the expected number of users in the system at any given point in time is 2.5λ. Thus, we could control the expected number of users by varying λ. For $|I| = 72$, we considered three scenarios, in which the expected number of users is 4, 6 or 8. For $|I| = 180$, we set the expected number of users to 10, 15 and 20. For $|I| = 300$, we set it 20, 30 and 40. This leads to nine scenarios in total (see the two left-most columns in Table 5.1).

In a similar manner, by careful selection of the scaling constant c, we could implicitly control the traffic intensity ρ. We considered four different values for c, corresponding to setting $\rho \in \{0.90, 0.95, 1.00, 1.05\}$. This means a total of 36 simulations. Each simulation was run for 1100 time periods, where the first 100 were used to allow the system to settle into steady state. So, we view T as being equal to 1000 in what follows.

5.5.2 Results

Recall that $J(t)$ denotes the set of users at time t and s_j denotes the satisfaction of user j. We first considered the following three performance measures.

- The *mean-min* satisfaction $\frac{1}{T}\sum_{t=1}^{T} \min_{j \in J(t)}\{s_j\}$.
- The *mean-mean* satisfaction $\frac{1}{T}\sum_{t=1}^{T} \frac{1}{|J(t)|}\sum_{j \in J(t)} s_j$.
- The *mean-max* satisfaction $\frac{1}{T}\sum_{t=1}^{T} \max_{j \in J(t)}\{s_j\}$.

We will call these simply "min", "mean" and "max" in what follows.

Table 5.1 shows the values taken by these three performance measures for various values of $|I|$ and various (expected) values of $|J|$, when $\rho = 1$. The columns headed "phase 1" concern a version of the heuristic in which the second improvement phase was omitted. We see that the heuristic performs remarkably well, with values close to or exceeding 1 in all cases. Interestingly, the second improvement phase has little effect on the mean satisfaction, but it improves the fairness of the solutions noticably. Note also that all three performance measures improve as the number of subcarriers increases, but worsen slightly as the expected number of users increases.

Table 5.2 reports the average time taken by each of our two improvement phases (i.e., Algorithms 4 and 5) in *one* time period, for the same simulations that were used

		Phase 1			Phase 2		
$\|I\|$	$\|J\|$	min	mean	max	min	mean	max
72	4	1.111	1.132	1.154	1.132	1.132	1.135
	6	1.055	1.094	1.133	1.090	1.094	1.105
	8	1.012	1.065	1.118	1.055	1.065	1.083
180	10	1.021	1.047	1.074	1.047	1.047	1.049
	15	0.983	1.025	1.067	1.023	1.025	1.034
	20	0.961	1.017	1.075	1.012	1.017	1.038
300	20	0.986	1.019	1.053	1.019	1.019	1.021
	30	0.960	1.012	1.065	1.009	1.012	1.029
	40	0.939	1.008	1.079	1.002	1.009	1.041

Table 5.1: Average values of performance measures during simulation ($\rho = 1$)

		Phase 1			Phase 2		
$\|I\|$	$\|J\|$	min	mean	max	min	mean	max
72	4	0.00005	0.00049	0.00207	0.00572	0.00811	0.01699
	6	0.00005	0.00077	0.00258	0.00590	0.00897	0.01860
	8	0.00005	0.00102	0.00311	0.00561	0.00898	0.01304
180	10	0.00014	0.00371	0.01154	0.04673	0.08529	0.12025
	15	0.00134	0.00620	0.01530	0.07202	0.09597	0.15561
	20	0.00206	0.00933	0.02383	0.08515	0.10873	0.17201
300	20	0.00528	0.01797	0.07447	0.30598	0.38317	0.91077
	30	0.01075	0.03051	0.10703	0.34284	0.42251	0.96528
	40	0.01781	0.04402	0.14996	0.34618	0.44667	1.09267

Table 5.2: Average values of running time ($\rho = 1$)

for Table 5.1. We see that, in most scenarios, the routine is extremely fast, taking less than 0.2 seconds. The exception is the case $|I| = 300$, for which phase 2 can take up to a second. This suggests that phase 2 may not be appropriate when one is dealing with a large base station.

Finally, we make some comments about the traffic intensity, ρ. As one might expect, changing the value of ρ affected all three performance measures. Interestingly, in all cases we tried, the net effect was simply to multiply each number in Table 5.1 by approximately $1/\rho$. As for running times, varying ρ had no noticeable effect. (This is probably because the bottleneck of the algorithm is phase 2, whose running time, $O(|I|^2)$, does not depend on ρ.) For these reasons, and also for the sake of brevity, we do not report detailed results for different values of ρ. In any case, the main

conclusion is that the performance of the heuristic is not very sensitive to the traffic intensity.

5.6 Conclusion

In this paper, we have considered how to allocate resources in an OFDMA system when (a) the system is overloaded (i.e., the expected demand is close to or higher than the system capacity), and (b) users arrive and depart every few seconds, in a stochastic manner. Since an exact approach for this case seems to be out of the question, we have proposed a dynamic local search heuristic. The computational results indicate that our heuristic consistently achieves allocations that are both efficient and fair.

Chapter 6

Conclusions and Future Work

6.1 Summary

Mobile wireless communications provides a rich and important source of challenging and fascinating optimisation problems. In this thesis, we have focused on joint subcarrier and power allocation problems in OFDMA systems. These particular problems are both combinatorial and non-linear, and most of them have been shown to be NP-hard ([32,52,53]). Moreover, in practice, one typically needs to compute solutions of good quality within a fraction of a second. For this reason, up to now, most research papers have focused on the development of fast heuristics for such problems. In this thesis, we have developed and tested some sophisticated exact algorithms for two problem variations, in addition to some new and effective heuristics.

In the first paper, presented in Chapter 2, we introduced the *joint subcarrier and power allocation problem with rate constraints* (SPARC). The purpose of the rate constraints is to ensure quality of service (QoS). To develop an exact algorithm for the SPARC problem, we added cutting planes generated by perspective functions, which we call perspective cuts, to reduce the feasible region of the continuous relaxation. By iteratively adding cutting planes, we obtained a tight polyhedron containing the feasible region. Our experiments show that this outer approximation method with perspective cuts can significantly reduce the running time of solving the SPARC problem. To further speed up our algorithm, we utilised three additional techniques: pre-emptive cuts, pre-processing and warm-starting. These techniques reduced both the number of iterations and running time by orders of magnitude. The exact algorithm we proposed is potentially of practical use, especially in a so-called *slow fading* environment, where users switch base stations infrequently.

We presented another variation problem in our second paper (Chapter 3). The *fractional subcarrier and power allocation with rate constraints* (F-SPARC) problem,

which is similar to the SPARC problem with the objective being to optimise energy efficiency instead of overall transmission rate. The change in objective function made the problem non-convex, thus more difficult to solve. We found that when dealing with such mixed-integer fractional programs with indicator variables, one needs to use bi-perspective reformulations and cuts in order to obtain bounds which are useful within an exact solution algorithm. We also believe that extensions of perspective reformulations and cuts to other classes of problems would be a valuable topic for future research.

In Chapter 4, we presented our third paper which proposes a heuristic algorithm for solving the F-SPARC problem more efficiently. The heuristic is based on a combination of fractional programming, 0-1 linear programming and binary search. The combination is remarkably effective, realistically solving within 1% of optimality, in a few seconds. We believe that our heuristic is suitable for real-life application in a slow-fading dynamic environment, where users arrive and depart only every few seconds.

Finally, our fourth paper in Chapter 5 considered the SPARC problem in dynamic overloaded cases where user demand often exceeds the system capacity. We addressed solving the SPARC problem in scenarios where users arrive and depart much more frequently. Furthermore, we also took fairness into consideration so that all users will receive transmission rates related to their demands. Our experiments show that our heuristic algorithm is both efficient and effective when dealing with realistic problems for small or medium base stations.

6.2 Suggestions for Future Work

In conducting our research, we formulated suggestions for further related research. The suggestions are as follows:

1. We assumed that the SPARC and F-SPARC are NP-hard, based on the fact that many similar problems associated with OFDMA systems have been shown to be NP-hard. A formal proof of hardness would however be desirable.

2. In the first paper, we used the OA method to solve the SPARC problem. We solve a series of MILPs, and therefore create and destroy several branch-and-bound trees along the way (see [5,63]). This could be wasting potentially useful information, as well as time. It would be interesting to design and code a more

sophisticated exact algorithm, in which there is only one branch-and-bound tree, and perspective cuts may be added to the subproblems within the tree.

3. We introduced pre-emptive cuts in Chapter 2 for symmetry breaking. Pre-emptive cuts are generated for all users once a cut has been generated for a specific subcarrier i and user j. This may result in a lot of redundant constraints and slower the solving process of the MILP relaxation. Therefore, one may try generating pre-emptive cuts only for users with similar demands.

4. In Chapter 3, we programmed our algorithm in Julia, which turned out to be a lot slower than C language. Also, due to precision issues in Julia, data for subcarrier noise are much higher than normal values. We believe coding the algorithm in C language will save running time and generate more accurate experiment results.

5. We used binary search in our third paper, presented in Chapter 4. We found that although the heuristic performed well, ranges of ϵ could be affected by random instances. In general, higher values of ϵ make the "NLP2(ϵ)" problem easier to solve. However, it also depends on the values of subcarrier noise and user demands. In bad cases during the binary search, the range of ϵ could be made too high due to some random infeasible "NLP2(ϵ)" problems. One could solve this problem by implementing a simulated annealing method inside the binary search. However, it should be noted this could lead to an increase in running time.

6. The heuristic method for the dynamic SPARC, that we described in Chapter 5, is suitable in practice only when the base station is relatively small (with, say, no more than 144 active subcarriers). This is due to the fact that, in many real-life settings, solutions are required within a fraction of a second. We are therefore interested in further research on more efficient heuristic algorithms, that can be used for instances with larger base stations (which, at the time of writing, can have up to 1440 active subcarriers).

Bibliography

[1] Gaussian channel. `https://www.eit.lth.se/fileadmin/eit/courses/eit080/InfoTheory`

[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh & P.H. Vance (1998) Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3), 316–329.

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

[4] J.F. Benders (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1), 238–252.

[5] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya *et al.* (2008) An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2), 186–204.

[6] P. Bonami, M. Kilinç & J. Linderoth (2012) Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*, 1–39. Springer.

[7] P. Bonami, A. Lodi, A. Tramontani & S. Wiese (2015) On mathematical programming with indicator constraints. *Mathematical programming*, 151(1), 191–223.

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

[9] S. Burer & A.N. Letchford (2012) Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2), 97–106.

[10] A. Charnes & W.W. Cooper (1962) Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4), 181–186.

[11] P.L. Combettes (2017) Perspective functions: Properties, constructions, and examples. *Set-Valued and Variational Analysis*, B28, 131.

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

[13] C. D'Ambrosio & A. Lodi (2013) Mixed integer nonlinear programming tools: an updated practical overview. *Annals of Operations Research*, 204(1), 301–320.

[14] G.B. Dantzig & P. Wolfe (1960) Decomposition principle for linear programs. *Operations research*, 8(1), 101–111.

[15] M.A. Duran & I.E. Grossmann (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3), 307–339.

[16] M. Ergen, S. Coleri & P. Varaiya (2003) QoS aware adaptive resource allocation techniques for fair scheduling in OFDMA based broadband wireless access systems. *IEEE Transactions on broadcasting*, 49(4), 362–370.

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

[18] R. Fletcher & S. Leyffer (1994) Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1), 327–349.

[19] C.A. Floudas & C.E. Gounaris (2009) A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1), 3–38.

[20] A. Frangioni, L. Galli & M.G. Scutellà (2015) Delay-constrained shortest paths: Approximation algorithms and second-order cone models. *Journal of Optimization Theory and Applications*, 164(3), 1051–1077.

[21] A. Frangioni, L. Galli & G. Stea (2015) Optimal joint path computation and rate allocation for real-time traffic. *The Computer Journal*, 58(6), 1416–1430.

[22] A. Frangioni & C. Gentile (2006) Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2), 225–236.

[23] 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 Networks and Applications*, 17(1), 29–35.

[24] R.E. Gomory *et al.* (1958) Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical society*, 64(5), 275–278.

[25] I.E. Grossmann (2002) Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, 3(3), 227–252.

[26] O. Günlük & J. Linderoth (2011) Perspective reformulation and applications. In *Mixed Integer Nonlinear Programming*, 61–89. New York, NY: Springer New York.

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

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

[29] R. Hemmecke, M. Köppe, J. Lee & R. Weismantel (2010) Nonlinear integer programming. In *50 Years of Integer Programming 1958-2008*, 561–618. Springer.

[30] J.B. Hiriart-Urruty & C. Lemaréchal (1993) Convex analysis and minimization algorithms.

[31] J.B. Hiriart-Urruty & C. Lemaréchal (2012) *Fundamentals of convex analysis*. Springer Science & Business Media.

[32] 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.

[33] K. Illanko, M. Naeem, A. Anpalagan & D. Androutsos (2014) Frequency and power allocation for energy efficient OFDMA systems with proportional rate constraints. *IEEE Wireless Communications Letters*, 3(3), 313–316.

[34] C. Isheden, Z. Chong, E. Jorswieck & G. Fettweis (2012) Framework for link-level energy efficiency optimization with informed transmitter. *IEEE Transactions on Wireless Communications*, 11(8), 1–12.

[35] R. Jain, D.M. Chiu & W.R. Hawe (1984) *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*, volume 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA.

[36] N. Karmarkar (1984) A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4), 373–395.

[37] R.M. Karp (1972) Reducibility among combinatorial problems. In *Complexity of computer computations*, 85–103. Springer.

[38] J. Kelley J (1960) The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4), 703–712.

[39] F.P. Kelly, A.K. Maulloo & D.K. Tan (1998) Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3), 237–252.

[40] L.G. Khachiyan (1979) A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244, 1093–1096.

[41] K. Kim, Y. Han & S.L. Kim (2005) Joint subcarrier and power allocation in uplink OFDMA systems. *IEEE Communications Letters*, 9(6), 526–528.

[42] D. Kivanc & H. Liu (2000) Subcarrier allocation and power control for OFDMA. In *Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on*, volume 1, 147–151. IEEE.

[43] V. Klee & G.J. Minty (1972) How good is the simplex algorithm? In O. Shisha (ed.) *Inequalities*, volume III, 159–175. New York: Academic Press.

[44] B. Korte & J. Vygen (2012) *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition.

[45] H.R.V. L. Transmission of information1. *Bell System Technical Journal*, 7(3), 535–563.

[46] A.H. Land & A.G. Doig (1960) An automatic method of solving discrete programming problems. *Econometrica*, 28(3), 497–520.

[47] 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.

[48] 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.

[49] 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.

[50] A.N. Letchford, Q. Ni & Z. Zhong (2018) A heuristic for maximising energy efficiency in an OFDMA system subject to QoS constraints. In J. Lee, G. Rinaldi & A.R. Mahjoub (eds.) *Combinatorial Optimization*, 303–312. Cham: Springer International Publishing.

[51] S. Leyffer, J. Linderoth, J. Luedtke, A. Miller & T. Munson (2009) Applications and algorithms for mixed integer nonlinear programming. In *Journal of Physics: Conference Series*, volume 180, 012014. IOP Publishing.

[52] 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.

[53] Z.Q. Luo & S. Zhang (2008) Dynamic spectrum management: complexity and duality. *IEEE Journal of Selected Topics in Signal Processing*, 2(1), 57–73.

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

[55] W.V.F. Maurício, F.R.M. Lima, D.A. Sousa, T.F. Maciel & F.R.P. Cavalcanti (2016) Joint resource block assignment and power allocation problem for rate maximization with QoS guarantees in multiservice wireless systems. *Journal of Communication and Information Systems*, 31(1).

[56] G.P. McCormick (1976) Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10(1), 147–175.

[57] H.D. Mittelmann (2018). Decision tree for optimization software. `http://plato.asu.edu/sub/nlores.html#mixedinteger`.

[58] J. Mo & J. Walrand (2000) Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5), 556–567.

[59] M.K. Muller, S. Schwarz & M. Rupp (2013) QoS investigation of proportional fair scheduling in lte networks. In *Wireless Days (WD), 2013 IFIP*, 1–4. IEEE.

[60] K.G. Murty (1983) Linear programming. *(1983)*.

[61] K.G. Murty & S.N. Kabadi (1987) Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2), 117–129.

[62] M. Padberg & G. Rinaldi (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1), 60–100.

[63] 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.

[64] M.G.C. Resende & P.M. Pardalos (eds.) (2006) *Handbook of Optimization in Telecommunications*. Boston, MA: Springer US.

[65] W. Rhee & J.M. Cioffi (2000) Increase in capacity of multiuser OFDM system using dynamic subchannel allocation. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 2, 1085–1089. IEEE.

[66] R.T. Rockerfeller (1970) Convex analysis.

[67] H.S. Ryoo & N.V. Sahinidis (1996) A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2), 107–138.

[68] S. Schaible (1974) Parameter-free convex equivalent and dual programs of fractional programming problems. *Zeitschrift für Operations Research*, 18(5), 187–196.

[69] S. Schaible (1983) Fractional programming. *Zeitschrift für Operations Research*, 27(1), 39–54.

[70] 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.

[71] C.E. Shannon (1949) Communication in the presence of noise. *Proceedings of the IRE*, 37(1), 10–21.

[72] C.E. Shannon (1949) Communication in the presence of noise. *Proceedings of the IRE*, 37(1), 10–21.

[73] Z. Shen, J.G. Andrews & B.L. Evans (2005) Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints. *IEEE transactions on wireless communications*, 4(6), 2726–2737.

[74] Z. Shen, J.G. Andrews & B.L. Evans (2005) Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints. *IEEE transactions on wireless communications*, 4(6), 2726–2737.

[75] Z. Song, Q. Ni, K. Navaie, S. Hou, S. Wu & X. Sun (2016) On the spectral-energy efficiency and rate fairness tradeoff in relay-aided cooperative OFDMA systems. *IEEE Transactions on Wireless Communications*, 15(9), 6342–6355.

[76] M. Tao, Y.C. Liang & F. Zhang (2008) Resource allocation for delay differentiated traffic in multiuser OFDM systems. *IEEE Transactions on Wireless Communications*, 7(6), 2190–2201.

[77] M. Tawarmalani & N.V. Sahinidis (2005) A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2), 225–249.

[78] B.W. Taylor III (2007) *Introduction to Management Science*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

[79] 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.

[80] 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.

[81] F. Trespalacios & I.E. Grossmann (2014) Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7), 991–1012.

[82] C.Y. Wong, R.S. Cheng, K.B. Lataief & R.D. Murch (1999) Multiuser OFDM with adaptive subcarrier, bit, and power allocation. *IEEE Journal on selected areas in communications*, 17(10), 1747–1758.

[83] C.Y. Wong, R.S. Cheng, K.B. Lataief & R.D. Murch (1999) Multiuser OFDM with adaptive subcarrier, bit, and power allocation. *IEEE Journal on selected areas in communications*, 17(10), 1747–1758.

[84] 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.

[85] 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.

[86] 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.

[87] W. Yu & R. Lui (2006) Dual methods for nonconvex spectrum optimization of multicarrier systems. *IEEE Transactions on Communications*, 54(7), 1310–1322.

[88] 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.