Monotonic Optimization in Communication and Networking Systems
By Ying Jun (Angela) Zhang, Liping Qian and Jianwei Huang

Contents

1 Introduction 2
1.1 Monotonic Optimization Theory and Applications 2
1.2 Outline 4
1.3 Notations 5

I Theory 6

2 Problem Formulation 7
2.1 Preliminary 7
2.2 Canonical Monotonic Optimization Formulation 10
2.3 Problems with Hidden Monotonicity 11
2.4 Monotonic Minimization Problem 15

3 Algorithms 16
3.1 An Intuitive Description 16
3.2 Basic Polyblock Outer Approximation Algorithm 18
3.3 Enhancements 26
3.4 Discrete Monotonic Optimization 31
II Applications

4 Power Control in Wireless Networks

4.1 System Model and Problem Formulation
4.2 Algorithm
4.3 Numerical Results

5 Power Controlled Scheduling in Wireless Networks

5.1 System Model and Problem Formulation
5.2 An Accelerated Algorithm
5.3 Numerical Results

6 Optimal Transmit Beamforming in MISO Interference Channels

6.1 System Model and Problem Formulation
6.2 Algorithm
6.3 Extensions

7 Optimal Random Medium Access Control (MAC)

7.1 System Model and Problem Formulation
7.2 Algorithm
7.3 Discussions

8 Concluding Remarks

References
Monotonic Optimization in Communication and Networking Systems

Ying Jun (Angela) Zhang¹, Liping Qian² and Jianwei Huang³

¹ Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, yjzhang@ie.cuhk.edu.hk
² College of Information Engineering, Zhejiang University of Technology, China, qianjoe@gmail.com
³ Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, jwhuang@ie.cuhk.edu.hk

Abstract

Optimization has been widely used in recent design of communication and networking systems. One major hurdle in this endeavor lies in the nonconvexity of many optimization problems that arise from practical systems. To address this issue, we observe that most nonconvex problems encountered in communication and networking systems exhibit monotonicity or hidden monotonicity structures. A systematic use of the monotonicity properties would substantially alleviate the difficulty in obtaining the global optimal solutions of the problems. This monograph provides a succinct and accessible introduction to monotonic optimization, including the formulation skills and solution algorithms. Through several application examples, we will illustrate modeling techniques and algorithm details of monotonic optimization in various scenarios. With this promising technique, many previously difficult problems can now be solved with great efficiency. With this monograph, we wish to spur new research activities in broadening the scope of application of monotonic optimization in communication and networking systems.
1

Introduction

1.1 Monotonic Optimization Theory and Applications

The global data traffic has reached 885 petabytes per month in 2012, which is more than ten times the global Internet traffic in the entire year of 2000. The rapid demand growth drives the research community to develop evolutionary and revolutionary approaches that push the communication and networking system performance toward new limits. To this end, optimization techniques have been proved extremely useful in approaching the utmost capacity of the limited available radio resources. Indeed, optimization methods have been successfully employed to obtain the optimal strategies for, for example, radio resource allocation, routing and scheduling, power control and interference avoidance, MIMO transceiver design, TCP flow control, and localization, just to name a few.

Most recent advances of optimization techniques rely crucially on the convexity of the problem formulation. Nonetheless, many problems encountered in practical engineering systems are nonconvex by their very nature. These problems are not only nonconvex in their original forms, but also cannot be equivalently transformed to convex ones by
1.1 Monotonic Optimization Theory and Applications

1 Note that there are also problems that are seemingly nonconvex, but can be equivalently transformed to convex problems by existing known methods, for example, change of variables. Such problems are NOT considered as nonconvex in our context.

2 $z - R_+^n$ and $z + R_+^n$ correspond to the sets $\{z'|z' \leq z\}$ and $\{z'|z' \geq z\}$, respectively.

any existing means. One such example is power control for throughput maximization in wireless networks. Another example is general utility maximization in random access networks.

An encouraging observation, however, is that a majority of nonconvex problems encountered in communication and networking systems exhibit monotonicity or hidden monotonicity structures. For example, the capacity and reliability of a wireless link monotonically increase with the bandwidth and SINR (signal to interference and noise ratio) of the link, and the quality of service provided to a user is a nondecreasing function of the amount of radio resources dedicated to the user. A systematic use of monotonicity properties may substantially alleviate the difficulty in obtaining the global optimal solution(s) of the problems, and this is indeed the key idea behind the monotonic optimization theory.

The theory of monotonic optimization has been established relatively recently by a series of papers by Tuy [22, 31, 32, 33, 34, 35, 37] and others [17, 27]. To intuitively understand the potential advantages offered by a monotonicity structure, recall that the search for a global optimal solution of a nonconvex optimization problem can involve examining every feasible point in the entire feasible region. If the objective function $f(z) : \mathcal{R}^n \to \mathcal{R}$ to be maximized is increasing, however, then once a feasible point $z$ is known, one can ignore the whole cone $z - R_+^n$ because no better feasible solution can be found in this cone. On the other hand, if the function $g(z) : \mathcal{R}^n \to \mathcal{R}$ is constraint like $g(z) \leq 0$ is increasing, then once a point $z$ is known to be infeasible, the whole cone $z + R_+^n$ can be ignored, because no feasible solution can be found in this cone. As such, the monotonic nature of the objective function and constraints allows us to limit the global search to a much smaller region of the feasible set, thus drastically simplifying the problem.

Only very recently was monotonic optimization introduced to the communication and networking research community. The first attempt
was made by Qian et al. [24], where the global optimal power control solution of ad hoc networks was found by exploiting the hidden monotonicity of the nonconvex power control problem. This work was subsequently followed by a number of researchers [5, 9, 12, 13, 15, 16, 19, 23, 25, 38, 39, 41, 43], where monotonicity or hidden monotonicity structures were exploited to solve a variety of nonconvex problems arising from areas including capacity maximization, scheduling, MIMO precoding and detection, distributed antenna coordination, and optimal relaying, etc. By and large, the application of monotonic optimization in communication and networking systems is still at its infancy stage, mainly because the technique is relatively new and unfamiliar to the communication and networking community. This is contrasted by the fact that most nonconvex problems considered in the communication and networking community are indeed monotonic.

The purpose of this monograph is to provide a succinct and accessible introduction to the theory and algorithms of monotonic optimization. Through several application examples, we will illustrate modeling techniques and algorithm details of monotonic optimization in various engineering scenarios. This is a humble attempt to spur new research activities in substantially broadening the scope of application of this promising technique in communication and networking systems.

1.2 Outline

There are two main parts in this monograph. Part I focuses on the theory and Part II on the application.

Part I consists of Sections 2 and 3, and is mainly based on the work of Tuy et al. [15, 22, 31, 32, 33, 34, 35, 37]. In particular, Section 2 discusses the formulation techniques, including the canonical formulation of monotonic optimization problems and problems that can be transformed into the canonical form. Section 3 introduces the polyblock outer approximation algorithm and its various enhancements that expedite the algorithm. The discussion is then extended to problems with discrete variables.

Part II consists of Sections 4–7. In particular, Section 4 discusses nonconvex power control in wireless interference channels,
problem formulations belong to a special class of monotonic optimization problems, namely, general linear fractional programming. Section 5 discusses power controlled scheduling problems, where we show how to reduce the variable size by exploiting the convexity of some set. Section 6 extends the discussion to multi-antenna systems where the objective is to optimize the transmitter beamforming. In this section we illustrate how to deal with vector variables in the polyblock outer approximation algorithm. Finally, Section 7 concerns network utility maximization in random access networks, where the problem is to maximize an increasing function of polynomials. Through this problem, we illustrate the use of auxiliary variables to convert a “difference of monotonic” optimization problem to a canonical monotonic optimization problem.

1.3 Notations

Throughout this monograph, vectors are denoted by boldface lower case letters and matrices are denoted by boldface upper case letters. The $i^{th}$ entry of a vector $x$ is denoted by $x_i$. We use $\mathcal{R}$, $\mathcal{R}_+$, and $\mathcal{R}_{++}$ to denote the set of real numbers, nonnegative real numbers, and positive real numbers, respectively. The set of $n$-dimensional real, nonnegative real, and positive real vectors are denoted by $\mathcal{R}^n$, $\mathcal{R}_+^n$, $\mathcal{R}_{++}^n$, respectively. $e^i \in \mathcal{R}^n$ denotes the $i^{th}$ unit vector of $\mathcal{R}^n$, i.e., the vector such that $e^i_i = 1$ and $e^i_j = 0$ for all $j \neq i$. $\mathbf{e} \in \mathcal{R}^n$ is an all-one vector.

For any two vectors $x, y \in \mathcal{R}$, we say $x \leq y$ (or $x < y$) if $x_i \leq y_i$ (or $x_i < y_i$) for all $i = 1, \cdots, n$. When $x \leq y$, we also say $y$ dominates $x$ or $x$ is dominated by $y$. Moreover, $x - \mathcal{R}^n_+$ and $x + \mathcal{R}^n_+$ correspond to the cones $\{x' | x' \leq x\}$ and $\{x' | x' \geq x\}$, respectively. $\cup$, $\cap$, and \ represent set union, set intersection, and set difference operators, respectively.
Part I

Theory
Problem Formulation

This section begins with a mathematical preliminary. The canonical monotonic maximization formulation is presented in Section 2.2. This is followed by Section 2.3 that demonstrates the technique to formulate problems with hidden monotonicity into the canonical form. Finally, we briefly discuss the monotonic minimization problems in Section 2.4.

2.1 Preliminary

Let us first introduce some definitions that will be useful later.

**Definition 2.1 (Increasing functions).** A function $f : \mathbb{R}^n_+ \to \mathbb{R}$ is increasing if $f(x) \leq f(y)$ when $0 \leq x \leq y$. A function $f$ is decreasing if $-f$ is increasing.

**Definition 2.2 (Boxes).** If $a \leq b$, then box $[a, b]$ is the set of all $x \in \mathbb{R}^n$ satisfying $a \leq x \leq b$. A box is also referred to as a hyper-rectangle.
Definition 2.3 (Normal sets). A set $G \subset \mathbb{R}_+^n$ is normal if for any point $x \in G$, all other points $x'$ such that $0 \leq x' \leq x$ are also in set $G$. In other words, $G \subset \mathbb{R}_+^n$ is normal if $x \in G \Rightarrow [0, x] \subset G$.

Definition 2.4 (Conormal sets). A set $H$ is conormal if $x \in H$ and $x' \geq x$ implies $x' \in H$. The set is conormal in $[0, b]$ if $x \in H \Rightarrow [x, b] \subset H$. Clearly, a set $H$ is conormal in $[0, b]$ if and only if the set $[0, b] \setminus H$ is normal.

Definition 2.5 (Normal hull). The normal hull of a set $A \subset \mathbb{R}_+^n$ is the smallest normal set containing $A$. Mathematically, the normal hull is given by $\mathcal{N}(A) = \bigcup_{z \in A} [0, z]$. Moreover, if $A$ is compact, so is $\mathcal{N}(A)$.

Definition 2.6 (Upper boundary). A point $\bar{x}$ of a normal closed set $G$ is called an upper boundary point of $G$ if $G \cap \{x \in \mathbb{R}_+^n | x > \bar{x}\} = \emptyset$. The set of all upper boundary points of $G$ is called its upper boundary and denoted by $\partial^+ G$.

To better understand the concepts, consider the example in Figure 2.1. Here, the rectangle represents box $[0, b]$. Set $H$ is a conormal set in box $[0, b]$. Its complement, i.e., $[0, b] \setminus H$, is set $G$ that is obviously a normal set. Meanwhile, $G$ is also the normal hull of the yellow set $A$. The red curve is the upper boundary of $G$, denoted by $\partial^+ G$.

Definitions 2.7 and 2.8 introduce the concepts of polyblocks, which are essential building blocks of the polyblock outer approximation algorithms that solve monotonic optimization problems.

Definition 2.7 (Polyblocks). A set $P \subset \mathbb{R}_+^n$ is called a polyblock if it is a union of a finite number of boxes $[0, z]$, where $z \in T$ and $|T| < +\infty$. The set $T$ is called the vertex set of the polyblock. A polyblock is clearly a normal set.
Definition 2.8 (Proper vertices of a polyblock). Let $\mathcal{T}$ be the vertex set of a polyblock $\mathcal{P} \subset \mathbb{R}_+^n$. A vertex $v \in \mathcal{T}$ is said to be proper if there is no $v' \in \mathcal{T}$ such that $v' \neq v$ and $v' \geq v$. A vertex is said to be improper if it is not proper. Improper vertices can be removed from the vertex set $\mathcal{T}$ without affecting the shape of the polyblock.

Figure 2.2 shows a polyblock with vertices $v_1$, $v_2$, and $v_3$. Here, $v_1$ and $v_2$ are proper vertices. In contrast, $v_3$ is an improper vertex and can be removed without affecting the polyblock. That is, the polyblock is the same as the one with proper vertices $v_1$ and $v_2$ only.
With the above definitions, we proceed to present the canonical formulation of monotonic optimization.

## 2.2 Canonical Monotonic Optimization Formulation

Monotonic Optimization is concerned with problems of the following form:

$$\max \{ f(x) | x \in G \cap H \}, \quad (2.1)$$

where $f(x) : \mathbb{R}^n_+ \to \mathbb{R}$ is an increasing function, $G \subset [0, b] \subset \mathbb{R}^n_+$ is a compact normal set with nonempty interior, and $H$ is a closed conormal set on $[0, b]$. Sometimes, $H$ is not present in the formulation, and the problem becomes

$$\max \{ f(x) | x \in G \}. \quad (2.2)$$

In this case, we can assume that the conormal set $H$ in Equation (2.1) is box $[0, b]$ itself. In the remaining of this monograph, we assume that the problem considered is feasible, i.e., $G \cap H \neq \emptyset$.

In real applications, sets $G$ and $H$ often result from constraints involving increasing functions $g_i(x) : \mathbb{R}^n_+ \to \mathbb{R}$ and $h_i(x) : \mathbb{R}^n_+ \to \mathbb{R}$

$$g_i(x) \leq 0, \quad i = 1, \ldots, m_1,$$

$$h_i(x) \geq 0, \quad i = m_1 + 1, \ldots, m.$$

Setting $g(x) = \max\{g_1(x), \ldots, g_{m_1}(x)\}$ and $h(x) = \min\{h_{m_1+1}(x), \ldots, h_m(x)\}$, the above inequalities are equivalent to

$$g(x) \leq 0, \quad h(x) \geq 0.$$

The following proposition connects the inequality constraints with $G$ and $H$ in Equation (2.1).

**Proposition 2.1.** For any increasing function $g(x)$ on $\mathbb{R}^n_+$, the set $G = \{ x \in \mathbb{R}^n_+ | g(x) \leq 0 \}$ is normal and it is closed if $g(x)$ is lower semicontinuous. Similarly, for any increasing function $h(x)$ on $\mathbb{R}^n_+$, the set $H = \{ x \in \mathbb{R}^n_+ | h(x) \geq 0 \}$ is conormal and it is closed if $h(x)$ is upper semicontinuous.
2.3 Problems with Hidden Monotonicity

In real systems, \( f(x) \) may correspond to some system performance, \( g(x) \) may correspond to some scarce resources that have limited availability, and \( h(x) \) may correspond to users’ satisfaction which has to reach a certain level. In general, the constraints derived from practical systems may result in an arbitrarily shaped feasible set instead of the nicely shaped one in Equation (2.1). The following proposition shows that this kind of problem can still be formulated into the canonical form, as long as \( f(x) \) is increasing.

**Proposition 2.2.** If \( A \) is an arbitrary nonempty compact set on \( \mathbb{R}_+^n \) and \( f(x) \) is an increasing function on \( \mathbb{R}_+^n \), then the problem

\[
\max \{ f(x) | x \in A \}
\]

is equivalent to

\[
\max \{ f(x) | x \in G \}
\]

where \( G = \mathcal{N}(A) \) is the normal hull of \( A \).

### 2.3 Problems with Hidden Monotonicity

Intuitively, many engineering problems encountered in practice have monotonicity structures one way or another. Not all of them can be straightforwardly expressed in the canonical form (Equation (2.1)). The monotonicity property is often “hidden”. This section discusses how to explore the hidden monotonicity of an optimization problem and transform it into the canonical form.

#### 2.3.1 Hidden Monotonicity in the Objective Function

Consider the problem

\[
\max \{ \phi(u(x)) | x \in D \},
\]

where \( D \subset \mathbb{R}^n \) is a nonempty compact set, \( \phi : \mathbb{R}_+^m \to \mathbb{R} \) is an increasing function, and \( u(x) = [u_1(x), \ldots, u_m(x)] \), \( u_i : D \to \mathbb{R}_+^+ \) are positive-valued continuous functions on \( D \).

The objective function of Equation (2.3) is not an increasing function of \( x \) in general. A widely known example of such problems is the General Linear Fractional Programming (GLFP) defined as follows.
**Definition 2.9 (GLFP).** An optimization problem belongs to the class of GLFP if it can be represented by the following formulation:

\[
\text{maximize } \phi \left( \frac{f_1(x)}{g_1(x)}, \ldots, \frac{f_m(x)}{g_m(x)} \right) \\
\text{variables } x \in \mathcal{D}
\]

where the domain \( \mathcal{D} \) is a nonempty polytope\(^1\) in \( \mathbb{R}^n \) (the \( n \)-dim real domain), functions \( f_1, \ldots, f_m, g_1, \ldots, g_m : \mathcal{D} \to \mathbb{R}^+ \) are positive-valued linear affine functions on \( \mathcal{D} \), and function \( \phi : \mathbb{R}_+^m \to \mathbb{R} \) is increasing on \( \mathbb{R}_+^m \).

Problem (2.3) is equivalent to \( \max \{ \phi(y) | y \in \mathcal{u}(\mathcal{D}) \} \), which can be further written as follows by Proposition 2.2:

\[
\max \{ \phi(y) | y \in \mathcal{G} \},
\]

where \( \mathcal{G} = \mathcal{N}(\mathcal{u}(\mathcal{D})) = \{ y \in \mathbb{R}_+^m | y \leq \mathcal{u}(x), x \in \mathcal{D} \} \). Since \( \mathcal{u}(x) \) is continuous on \( \mathcal{D} \), \( \mathcal{u}(\mathcal{D}) \) is compact. Thus, its normal hull \( \mathcal{G} \) is also compact and is contained in box \([0, b]\). Furthermore, since \( u_i(x) \)'s are positive, \( \mathcal{G} \) has a nonempty interior. By this, we conclude that Equation (2.5) is a monotonic optimization problem in the canonical form.

### 2.3.2 Hidden Monotonicity in the Constraint

Consider the problem

\[
\max \{ f(x) | x \in \mathcal{D}, \phi(u(x)) \leq 0 \},
\]

where \( \mathcal{D}, \phi, \) and \( u \) are defined as previously, and \( f(x) \) is a continuous function. Here, the feasible set is not defined by increasing functions of \( x \), and hence is not normal in \( x \). To explore the hidden monotonicity, note that the following set is closed and conormal due to the continuity of \( u \):

\[ \mathcal{H} = \{ y \in \mathbb{R}_+^m | u(x) \leq y, x \in \mathcal{D} \}. \]

---

\(^1\)Polytope means the generalization to any dimension of polygon in two dimensions, polyhedron in three dimensions, and polychoron in four dimensions.
We can then rewrite Equation (2.6) as
\[
\max \{ \theta(y) | y \in \mathcal{H}, \phi(y) \leq 0 \},
\]
where
\[
\theta(y) = \begin{cases} 
\sup \{ f(x) | u(x) \leq y, x \in \mathcal{D} \}, & \text{if } y \in \mathcal{H}, \\
-M, & \text{otherwise}.
\end{cases}
\]
(2.8)

Here, \( M > 0 \) is an arbitrary number such that \(-M < \min \{ f(x) | x \in \mathcal{D} \} \).
If \( f(x) \) is concave, \( u_i(x) \)’s are convex, and \( \mathcal{D} \) is convex, then \( \theta(y) \) is the optimal value of a convex program.

**Proposition 2.3.** \( \theta(y) \) is increasing and upper semicontinuous on \( \mathbb{R}^m_+ \).

*Sketch of proof:* To see that \( \theta(y) \) is increasing, note that \( \{ x \in \mathcal{D} | u(x) \leq y \} \subset \{ x \in \mathcal{D} | u(x) \leq y' \} \) for any \( y \leq y' \) with \( y \in \mathcal{H} \). If \( y \leq y' \) and \( y \notin \mathcal{H} \), then by definition \( \theta(y') \geq \theta(y) = -M \). Moreover, the continuity of \( \theta(y) \) can be proved from the continuity of \( f(x) \) and \( u(x) \) and the compactness of \( \mathcal{D} \).

With Proposition 2.3 we can say Problem (2.7) is a monotonic optimization problem in the canonical form (2.1), with \( \mathcal{G} = \{ y \in \mathbb{R}^n_+ | \phi(y) \leq 0 \} \).

### 2.3.3 Maximization of Differences of Increasing Functions

Consider the problem
\[
\max \{ f(x) - g(x) | x \in \mathcal{G} \cap \mathcal{H} \},
\]
(2.9)
where \( f(x) \) and \( g(x) \) are increasing functions on \( \mathbb{R}^n_+ \). \( \mathcal{G} \) and \( \mathcal{H} \) are defined as in Equation (2.1). Note that Equation (2.9) captures a large class of problems. For example, any polynomial \( p(x) \) on \( \mathbb{R}^n_+ \) can be expressed as a difference of two increasing functions. This can be done by grouping separately the terms with positive coefficients and those with negative coefficients, and rewrite \( p(x) \) as \( p(x) = p_1(x) - p_2(x) \), where \( p_1(x) \) and \( p_2(x) \) are polynomials with positive coefficients, and hence are increasing functions.
Problem Formulation

To write Equation (2.9) in the canonical form, notice that for every $x \in [0, b]$ we have $g(x) \leq g(b)$. In other words, there exists a $t \geq 0$ such that $g(x) + t = g(b)$. Hence, the problem can be rewritten as

$$\max \{ f(x) + t - g(b) | x \in G \cap H, t = g(b) - g(x) \}.$$ 

Adding the constant $g(b)$ to the objective function, we obtain the problem

$$\max \{ f(x) + t | x \in G \cap H, t + g(x) = g(b) \},$$

which is equivalent to

$$\max \{ f(x) + t | x \in G \cap H, t + g(x) \leq g(b), 0 \leq t \leq g(b) - g(0) \},$$

(2.10)

since $f(x) + t$ and $t + g(x)$ are increasing.

Define $F(x, t) = f(x) + t$, and

$$\mathcal{D} = \{(x, t) | x \in G, t + g(x) \leq g(b), 0 \leq t \leq g(b) - g(0) \},$$

$$\mathcal{E} = \{(x, t) | x \in H, 0 \leq t \leq g(b) - g(0) \}.$$

It is easy to see that $F(x, t)$ is an increasing function on $\mathbb{R}^{n+1}_+$, $\mathcal{D}$ is a closed normal set contained in box $[0, b] \times [0, g(b) - g(0)]$, and $\mathcal{E}$ is a closed conormal set in this box. Hence, Problem (2.10) reduces to

$$\max \{ F(x, t) | (x, t) \in \mathcal{D} \cap \mathcal{E} \},$$

which is in the canonical form.

2.3.4 Difference of Increasing Functions in the Constraints

Finally, we consider the problem

$$\max \{ f(x) | g(x) - h(x) \leq 0, x \in \Omega \subset \mathbb{R}^n_+ \},$$

(2.11)

where $f$, $g$, and $h$ are increasing and continuous functions on $\mathbb{R}^n_+$, and $\Omega$ is a normal set contained in $[0, b] \subset \mathbb{R}^n_+$. To transform the problem into the canonical form, we can split the inequality $g(x) - h(x) \leq 0$ for $x \in \Omega$ into two inequalities:

$$g(x) + t \leq g(b), \quad h(x) + t \geq g(b),$$
where \( t \geq 0 \). Define
\[
\mathcal{G} = \{(x,t) \in \mathbb{R}_+^n \times \mathbb{R}_+ | x \in \Omega, g(x) + t \leq g(b), 0 \leq t \leq g(b) - g(0)\}
\]
\[
\mathcal{H} = \{(x,t) \in \mathbb{R}_+^n \times \mathbb{R}_+ | h(x) + t \geq g(b)\}.
\]
We can rewrite Equation (2.11) as
\[
\max \{f(x) | \mathcal{G} \cap \mathcal{H}\}.
\] (2.12)
Here, \( \mathcal{G} \) is a normal set contained in box \([0,b] \times [0,g(b) - g(0)]\) and \( \mathcal{H} \) is a conormal set. Thus, the problem is monotonic optimization in the canonical form.

### 2.4 Monotonic Minimization Problem

There is another class of monotonic optimization problems that minimize an increasing function. Consider, for example, the following problem
\[
\min \{f(x) | g(x) \leq 0, h(x) \geq 0, x \in [0,b]\},
\] (2.13)
where \( f(x) : \mathbb{R}_+^n \to \mathbb{R} \) is an increasing function. With some manipulations, the problem can be easily transformed to a monotonic maximization problem as follows:
\[
\max \{\tilde{f}(y) | \tilde{h}(y) \leq 0, \tilde{g}(y) \geq 0, y \in [0,b]\},
\] (2.14)
where \( \tilde{f}(y) = -f(b - y), \tilde{g}(y) = -g(b - y) \), and \( \tilde{h}(y) = -(b - y) \) are increasing functions on \([0,b]\).

With the above discussions, we will focus in this monograph monotonic maximization problems only.
3

Algorithms

3.1 An Intuitive Description

To solve a monotonic optimization problem, i.e., to maximize an increasing function over a normal set, it turns out that one can exploit a “separation property” of normal sets, which is analogous to the separation property of convex sets. It is well known that any point \( z \) outside a convex set can be strictly separated from the set by a half space. As a result, a convex feasible set can be approximated, as closely as desired, by a nested sequence of polyhedrons. Likewise, any point \( z \) outside a normal set is separated from the normal set by a cone congruent to the nonnegative orthant. Thus, a normal set can be approximated as closely as desired by a nested sequence of “polyblocks”. This is illustrated in Figure 3.1.

The separation property of convex sets plays a fundamental role in polyhedral outer approximation methods, which solve convex maximization problems over convex feasible sets. In particular, a convex function always achieves its maximum over a bounded polyhedron at one of its vertices. The well-studied polyhedral outer approximation algorithm successively maximizes the convex objective function on a
3.1 An Intuitive Description

Fig. 3.1 (a) A polyhedron enclosing a convex set. (b) A polyblock enclosing a normal set.

sequence of polyhedra that enclose the feasible set. At each iteration, the current enclosing polyhedron is shrunk by adding a cutting plane tangential to the feasible set at a boundary point \[10\].

Similarly, monotonic optimization problems can be solved by polyblock outer approximation algorithms that are analogous to, but not quite same as, polyhedral outer approximation algorithms. In particular, a monotonically increasing function always achieves its maximum over a polyblock at one of its vertices. The polyblock outer approximation algorithm successively maximizes the increasing objective function on a sequence of polyblocks that enclose the feasible set (or a subset of the feasible set that contains the optimal solution). At each iteration, the current enclosing polyblock is refined by cutting off a cone congruent to the nonnegative orthant.

We would like to emphasize that no existing algorithm can claim to solve a general nonconvex optimization problem efficiently within polynomial time, and the polyblock outer approximation algorithm is no exception. However, by exploiting the special structure of the problems, the computational complexity involved in solving the problems is much more manageable than generic algorithms. Given this said, when modeling a real-world problem into a monotonic optimization problem, one should cautiously reduce the dimension of the problem as much as possible to enable fast solution algorithms. Very often, the dimension reduction techniques are problem specific and require the domain knowledge of the underlying system. This will be further demonstrated in Sections 5 and 7 using the examples of power controlled scheduling and random medium access.
3.2 Basic Polyblock Outer Approximation Algorithm

Before getting into the details, we first introduce a few propositions that lie in the foundation of the polyblock outer approximation algorithms. These propositions will then be illustrated through a simple example in Figure 3.2.

**Proposition 3.1.** Let $\mathcal{G}$ be a compact normal set and $\mathcal{H}$ be a closed conormal set. The maximum of an increasing function $f(x)$ over $\mathcal{G} \cap \mathcal{H}$ is attained on $\partial^+ \mathcal{G} \cap \mathcal{H}$.

**Proposition 3.2.** The maximum of an increasing function $f(x)$ over a polyblock in $\mathbb{R}_n^+$ is attained at one of its proper vertices.

**Proposition 3.3 (Projection on the upper boundary).** Let $\mathcal{G} \subset \mathbb{R}_n^+$ be a compact normal set with nonempty interior. Then, for any point $z \in \mathbb{R}_n^+ \setminus \mathcal{G}$, the line segment joining $0$ to $z$ meets the upper boundary $\partial^+ \mathcal{G}$ of $\mathcal{G}$ at a unique point $\pi_\mathcal{G}(z)$, which is defined as

$$\pi_\mathcal{G}(z) = \lambda z, \quad \lambda = \max\{\alpha > 0 | \alpha z \in \mathcal{G}\}. \quad (3.1)$$

We call $\pi_\mathcal{G}(z)$ the projection of $z$ on the upper boundary of $\mathcal{G}$.

**Proposition 3.4 (Separation property of normal sets).** Let $\mathcal{G}$ be a compact normal set in $\mathbb{R}_n^+$ with nonempty interior, and $z \in \mathbb{R}_n^+ \setminus \mathcal{G}$. If $\bar{x} \in \partial^+ \mathcal{G}$ such that $\bar{x} < z$, then the cone $K_{\bar{x}} := \{x \in \mathbb{R}_n^+ | x > \bar{x}\}$ separates $z$ strictly from $\mathcal{G}$.

Corollary 3.5 is a straightforward result from Propositions 3.3 and 3.4.

**Corollary 3.5.** A point $z \in \mathbb{R}_n^+ \setminus \mathcal{G}$ is strictly separated from $\mathcal{G}$ by the cone $K_{\pi_\mathcal{G}(z)} := \{x \in \mathbb{R}_n^+ | x > \pi_\mathcal{G}(z)\}$. 
3.2 Basic Polyblock Outer Approximation Algorithm

Proposition 3.6. Let $\mathcal{P} \subset \mathbb{R}^n_+$ be a polyblock and $y \in \mathcal{P}$. Then, $\mathcal{P} \setminus \mathcal{K}_y^+$ is another polyblock.

The basic idea of polyblock outer approximation algorithm is to enclose the feasible set $\mathcal{G} \cap \mathcal{H}$ by a polyblock $\mathcal{P}_1$, as illustrated in Figure 3.2(a). Due to Proposition 3.2, the search for the global optimal solution over $\mathcal{P}_1$ reduces to choosing the best one among all of its proper vertices. Without loss of generality, let us say the optimal vertex is $v_1$. According to Proposition 3.3, we can find the projection of $v_1$ on the upper boundary of $\mathcal{G}$, denoted as $\pi_\mathcal{G}(v_1)$. Corollary 3.5 says cutting off the cone $\mathcal{K}^+_{\pi_\mathcal{G}(v_1)} := \{x \in \mathbb{R}^n_+ | x > \pi_\mathcal{G}(v_1)\}$ from $\mathcal{P}_1$ will not exclude...
any points in $G$. The resulting set $P_1 \setminus K_{\pi_G(v_1)}^+$ is still an enclosing polyblock due to Proposition 3.6 as illustrated in Figure 3.2(b). We denote it as $P_2$. Following this procedure, we can construct a sequence of polyblocks outer approximating the feasible set:

$$P_1 \supset P_2 \supset \cdots \supset P_k \supset \cdots \supset G \cap H$$

in such a way that

$$\max\{ f(x) | x \in P_k \} \searrow \max\{ f(x) | x \in G \cap H \}.$$

Here, $\searrow$ means converge from above, because the maximum of the increasing function over the feasible set $G \cap H$ is always no larger than that over the polyblock $P_k$ that encloses the feasible set.

We can further refine the enclosing polyblocks by removing the vertices that are not in $H$. The following proposition says that the resultant polyblock still encloses $G \cap H$.

**Proposition 3.7.** Let $P \subset \mathbb{R}^n_+$ be a polyblock enclosing $G \cap H$, where $G \subset \mathbb{R}^n_+$ is a bounded normal set and $H \subset \mathbb{R}^n_+$ is a closed conormal set, and let $T$ be the vertex set of $P$. Let $P'$ be another polyblock with vertex set $T \setminus \{ v \in T | v \notin H \}$. Then, $P' \supset G \cap H$.

Proposition 3.7 is illustrated in Figure 3.2(c). Here, $P'_2$ is obtained from $P_2$ in Figure 3.2(b) by removing the vertices that are not in $H$. Obviously, $P'_2$ still encloses $G \cap H$.

One may infer from the above description that there are three key ingredients in the polyblock outer approximation algorithm, namely, (i) computing the boundary point $\pi_G(v)$, (ii) generating the new polyblock from the old one, and (iii) terminating the algorithm when it converges to the optimal solution. In the following, we will discuss these three operations in detail. The convergence of the algorithm is discussed in Subsection 3.2.4.

### 3.2.1 Computing the Upper Boundary Point $\pi_G(z_k)$

Let $z_k$ denote the vertex of $P_k$ that maximizes the objective function $f$ over $P_k$. According to the definition in Equation (3.1), finding $\pi_G(z_k)$
Algorithm 1 Bisection Search to Compute the Upper Boundary Point \( \pi_G(z_k) \)

**Input:** \( z_k, G \)

**Output:** \( \alpha \) such that \( \alpha = \arg \max \{ \alpha > 0 | \alpha z_k \in G \} \)

1. Initialization: Let \( \alpha_{\text{min}} = 0 \) and \( \alpha_{\text{max}} = 1 \). Let \( \delta > 0 \) be a small positive number.
2. repeat
3. Let \( \bar{\alpha} = (\alpha_{\text{min}} + \alpha_{\text{max}})/2 \).
4. Check if \( \bar{\alpha} \) is feasible, i.e., if \( \bar{\alpha} z_k \in G \). If yes, let \( \alpha_{\text{min}} = \bar{\alpha} \). Else, let \( \alpha_{\text{max}} = \bar{\alpha} \).
5. until \( \alpha_{\text{max}} - \alpha_{\text{min}} \leq \delta \)
6. Output \( \alpha = \alpha_{\text{min}} \).

involves solving a one-dimensional optimization problem

\[
\max \{ \alpha > 0 | \alpha z_k \in G \} = \max \{ \alpha > 0 | g_1(\alpha z_k) \leq 1, g_2(\alpha z_k) \leq 1, \ldots \},
\]

where \( g_i(\cdot) \)'s are the increasing functions defining the normal set \( G \), as discussed at the beginning of Section 2. The computational complexity required in solving the problem depends heavily on the forms of \( g_i(\cdot) \). The computation becomes even trickier when hidden monotonicity is involved. Take the case in Section 2.3.1 for example. Finding \( \pi_G(z_k) \) reduces to solving a max–min problem as follows.

\[
\max \{ \alpha | \alpha z_k \leq u(x), x \in D \} = \max \{ \alpha | \alpha \leq \min_{i=1,\ldots,m} \frac{u_i(x)}{z_{ki}}, x \in D \}
= \max_{x \in D} \min_{i=1,\ldots,m} \frac{u_i(x)}{z_{ki}}. \tag{3.2}
\]

Problem (3.2) is convex if \( u_i(x) \)'s are quasiconcave in \( x \).

In general, due to the normality of \( G \), \( \alpha \) can be found by the bisection search algorithm described in Algorithm 1. The main operation here is the feasibility check in Line 4, the complexity of which again relies on the structure of \( G \). In Part II of this monograph, we will illustrate, through real-world applications, the techniques of efficiently computing \( \alpha \) in different scenarios.
3.2.2 Generating the New Polyblock

This subsection concerns the derivation of a new enclosing polyblock \( P_{k+1} \) from the old one \( P_k \) by cutting off a cone that is in the infeasible set. First, let us discuss the way to compute the proper vertex set of the polyblock \( P \setminus K^+ \), where \( K^+ \) is defined in Proposition 3.4.

Let \( T \) be the proper vertex set of polyblock \( P \). Then, \( T_* = \{ v \in T \mid v > x \} \) is the subset of \( T \) that contains all vertices that are in \( K^+ \). Also, for each vertex \( v \in T_* \), let us define \( v^i = v + (x_i - v_i)e_i \) for \( i = 1, \ldots, n \). Note that \( v^i \) is obtained by replacing the \( i^{th} \) entry of \( v \) by the \( i^{th} \) entry of \( x \). Take Figure 3.2(b) as an example. \( v_1^1 \) is obtained by replacing the 1\( st \) entry in \( v_1 \) with that of \( \pi_G(z_k) \), and \( v_1^2 \) is obtained by replacing the 2\( nd \) entry in \( v_1 \) with that of \( \pi_G(v_1) \).

**Proposition 3.8.** Let \( P \subseteq \mathcal{R}^n_+ \) be a polyblock with a proper vertex set \( T \subseteq \mathcal{R}^n_+ \) and let \( x \in P \). Then, the polyblock \( P \setminus K^+_x \) has a vertex set

\[
T' = (T \setminus T_*) \cup \{ v^i = v + (x_i - v_i)e_i \mid v \in T_*, i \in \{1, \ldots, n\} \}. \tag{3.3}
\]

The improper vertices in \( T' \) are those \( v^i = v + (x_i - v_i)e_i \) that are dominated by another vertex in \( T' \). By removing the improper vertices from \( T' \), we obtain the proper vertex set of the polyblock \( P \setminus K^+_x \).

Following the above procedure, we can generate \( P_{k+1} \) from \( P_k \) by cutting off the cone \( K^+_{\pi_G(z_k)} \).

3.2.3 Termination Criterion

Recall that \( z_k \) denotes the optimal vertex that maximizes \( f \) among all vertices of \( P_k \). For example, in Figure 3.2(a), \( z_1 = v_1 \). As \( P_k \) encloses the feasible set \( G \cap H, f(z_k) \geq f(x^*) \), where \( x^* \) is the optimal solution to Problem (2.1). Intuitively, we can terminate the algorithm when \( |f(z_k) - f(x)| \) is sufficiently small, where \( x \) is a best feasible solution known to us, and claim \( x \) to be the optimal solution. Note that it is sufficient to focus on the upper boundary points when searching for the optimal solution due to Proposition 3.1.

The above procedure can be refined as follows. Let \( \bar{x}_k \) denote the best feasible solution known so far at iteration \( k \), and
3.2 Basic Polyblock Outer Approximation Algorithm

$CBV_k = f(\bar{x}_k)$ denote the current best value. At the $k+1^{th}$ iteration, let $\bar{x}_{k+1} = \pi_G(z_{k+1})$ and $CBV_{k+1} = f(\pi_G(z_{k+1}))$ if $\pi_G(z_{k+1}) \in G \cap H$ and $f(\pi_G(z_{k+1})) \geq CBV_k$. Otherwise, let $\bar{x}_{k+1} = \bar{x}_k$ and $CBV_{k+1} = CBV_k$. The algorithm terminates when $|f(z_k) - CBV_k| \leq \epsilon$, where $\epsilon \geq 0$ is a given tolerance. We can show that an $\epsilon$-optimal solution can be obtained following this procedure.

Alternatively, we can terminate the algorithm when $z_k$ is sufficiently close to the feasible set, i.e., $|z_k - \bar{x}_k| \leq \delta$, where $\delta$ is a small positive number. The resultant solution is called a $\delta$-approximate optimal solution. As we will discuss later, this additional termination criterion is needed to guarantee the execution time of the algorithm to be finite.

With the above building blocks, we now summarize the polyblock approximation algorithm in Algorithm 2. In particular, Line 4 finds the vertex that maximizes the objective function value among all vertices of $P_k$. Line 5 computes the projection $\pi_G(z_k)$. Lines 6 and 7 indicate that the optimal solution is obtained if the optimal vertex $z_k$ is already in the feasible set. Otherwise, Lines 10 and 11 generate a smaller polyblock $P_{k+1}$ from $P_k$ by excluding the cone $K^+_{\pi_G(z_k)}$ and removing improper vertices and vertices that are not in $H$. The resulting $P_{k+1}$ still encloses $G \cap H$. Finally, the algorithm terminates when $f(z_k) - CBV_k$ is sufficiently small.

3.2.4 Convergence of the Polyblock Outer Approximation Algorithm

The convergence of the polyblock outer approximation algorithm can be proved under mild assumptions, namely, $f(x)$ is upper semicontinuous, $G$ has a nonempty interior, and $G \cap H \subset R^{n}_{++}$.

**Proposition 3.9.** With the above conditions, each of the generated sequences $\{z_k\}$ and $\{\bar{x}_k\}$ contains a subsequence converging to an exact optimal solution if Algorithm 2 is infinite. Moreover, if $f(x)$ is Lipschitz continuous, then Algorithm 2 is guaranteed to converge to an $\epsilon$-optimal solution in a finite number iterations for any given $\epsilon > 0$.

---

1 $z_k$ is said to be an $\epsilon$-optimal solution if $f(x^*) - \epsilon \leq f(z_k) \leq f(x^*)$. 
Algorithm 2 Polyblock Outer Approximation Algorithm

Input: An upper semicontinuous increasing function \( f(\cdot) : \mathbb{R}_+^n \to \mathbb{R} \)

\[ g \subset \mathbb{R}_+^n, \text{ and a closed conormal set } \mathcal{H} \subset \mathbb{R}_+^n \]

such that \( g \cap \mathcal{H} \neq \emptyset \)

Output: an \( \epsilon \)-optimal solution \( x^* \)

1: Initialization: Let the initial polyblock \( P_1 \) be box \([0,b]\) that encloses \( g \cap \mathcal{H} \). The vertex set \( T_1 = \{b\} \). Let \( \epsilon \geq 0 \) be a small positive number. \( CBV_0 = -\infty. \) \( k = 0. \)

2: repeat

3: \( k = k + 1. \)

4: From \( T_k \), select \( z_k \in \arg \max \{ f(z) | z \in T_k \}. \)

5: Compute \( \pi_G(z_k) \), the projection of \( z_k \) on the upper boundary of \( g \).

6: if \( \pi_G(z_k) = z_k \), i.e., \( z_k \in g \) then

7: \( \bar{x}_k = z_k \) and \( CBV_k = f(z_k) \).

8: else

9: If \( \pi_G(z_k) \in g \cap \mathcal{H} \) and \( f(\pi_G(z_k)) \geq CBV_{k-1} \), then let the current best solution \( \bar{x}_k = \pi_G(z_k) \) and \( CBV_k = f(\pi_G(z_k)) \). Otherwise, \( \bar{x}_k = \bar{x}_{k-1} \) and \( CBV_k = CBV_{k-1} \).

10: Let \( x = \pi_G(z_k) \) and

\( T_{k+1} = (T_k \setminus T_s) \cup \{v^i = v + (x_i - v_i)e^i | v \in T_s, i \in \{1,\ldots,n\}\}, \)

where \( T_s = \{v \in T_k | v > x\} \).

11: Remove from \( T_{k+1} \) the improper vertices and the vertices \( \{v \in T_{k+1} | v \notin \mathcal{H}\} \).

12: end if

13: until \( |f(z_k) - CBV_k| \leq \epsilon. \)

14: Let \( x^* = \bar{x}_k \) and terminate the algorithm.

Interested readers are referred to [31, 33] for the proof of the proposition.

We would like to comment that even if \( f(x) \) is not Lipschitz continuous, Algorithm 2 can still terminate in a finite number of iterations if an additional termination condition \( |z_k - \bar{x}_k| \leq \delta \) is added to Line 9 of the
3.2 Basic Polyblock Outer Approximation Algorithm

The resultant solution is either $\epsilon$-optimal or $\delta$-approximate optimal.

Let us now turn to discuss the convergence condition $\mathcal{G} \cap \mathcal{H} \subset \mathbb{R}^{n}_{++}$. This condition basically says that the conormal set $\mathcal{H}$ should be strictly bounded away from 0, namely, there exists a positive vector $a$ such that

$$0 < a \leq x, \ \forall x \in \mathcal{H}.$$  

When this condition does not hold, e.g., when $\mathcal{H}$ includes vectors with some entries being 0 (i.e., the case in Figure 3.3(a)), the convergence of Algorithm 2 cannot be guaranteed.

To restore this condition, one can shift the origin to the negative orthant, say to $-\beta e$, where $\beta > 0$ is chosen to be not too small (see Figure 3.3(b)). Then, with respect to the new origin, we have $\hat{\mathcal{H}} = \mathcal{H} + \beta e$ being the shifted conormal set and $\hat{\mathcal{G}} = \mathbb{R}^{n}_{+} \cap (\mathcal{G} + \beta e - \mathbb{R}^{n}_{+})$ being the normal hull of $\mathcal{G}$. Let

$$\hat{f}(x) = \begin{cases} f(x - \beta e), & \text{if } x \geq \beta e \\ -M, & \text{otherwise} \end{cases}$$

where $M > 0$ is a sufficiently large number. The original problem is equivalent to

$$\max\{\hat{f}(x) | x \in \hat{\mathcal{G}} \cap \hat{\mathcal{H}}\},$$

where $\hat{\mathcal{H}}$ is strictly bounded away from 0. As such, the convergence of the algorithm can be guaranteed. This technique will be illustrated through two examples in Sections 4 and 5.

![Fig. 3.3 Shift of origin.](image-url)
3.3 Enhancements

In this section, we discuss two enhancements that expedite the polyblock outer approximation algorithm. Subsection 3.3.1 is about reducing the vertex set $T_k$ at each iteration by removing unnecessary vertices, and Subsection 3.3.2 is about expediting the shrinking of the enclosing polyblock.

3.3.1 Removing Suboptimal Vertices

The size of the vertex set $T_k$ can grow quite large when $k$ is large. This not only leads to a high computational complexity to find the optimal vertex $z_k$, but also may cause memory overflow problems. On the other hand, many of the vertices are not needed in the computation, and therefore can be safely discarded. For example, we can discard the vertices that are obviously not optimal, i.e., vertices with values smaller than $CBV_k$. This is because every point that is smaller than these vertices would have even smaller values, and thus eliminating it will not exclude any optimal solutions.

With this, Lines 11 and 13 of Algorithm 2 are replaced by

11: Remove from $T_{k+1}$ the improper vertices and all vertices such that $\{v \in T_{k+1} | v \not\in H \text{ or } f(v) \leq CBV_k + \epsilon}\}$.
13: until $T_{k+1} = \emptyset$.

Hence, the sequence of enclosing polyblocks satisfies

$$\mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \cdots \supseteq \mathcal{P}_k \supseteq \mathcal{P}_{k+1} \supseteq \{x \in \mathcal{G} \cap H | f(x) > CBV_k + \epsilon\}.$$ 

3.3.2 Tightening the Enclosing Polyblock

The basic idea of the polyblock outer approximation algorithm is to approximate the feasible set (or a subset of the feasible set that contains optimal solutions) by enclosing polyblocks. Intuitively, the tighter an enclosing polyblock, the better it approximates the upper boundary of the feasible set. In this subsection, we discuss the derivation of a tighter polyblock $\mathcal{P}'$ from an original polyblock $\mathcal{P}$ such that

$$\mathcal{G} \cap H \cap \mathcal{P} \subset \mathcal{P}' \subset \mathcal{P}.$$
That is, \( P' \) is smaller than \( P \) and yet contains all points in \( G \cap H \) that were contained by \( P \).

As a simple example, let us first consider box \([0, b]\) that contains \( G \cap H \) or a part of it. We wish to find the smallest \( b' \) such that \([0, b']\) still contains \( G \cap H \cap [0, b] \).

**Proposition 3.10.** The smallest \( b' \) such that \([0, b']\) contains \( G \cap H \) is given by the following:

\[
b'_i = \max \{ x_i | x \in G \cap H \}, \quad \forall i = 1, \ldots, n.
\]

Likewise, given box \([0, b]\), the smallest \( b' \) such that \([0, b']\) contains \( G \cap H \cap [0, b] \) is given by the following:

\[
b'_i = \max \{ x_i | x \in G \cap H \cap [0, b] \}, \quad \forall i = 1, \ldots, n.
\]

The above procedure is illustrated in Figure 3.4 where the box is tightened by cutting off half spaces that do not contain the set \( G \cap H \cap [0, b] \).

Now we tighten a polyblock \( P \) that contains \( G \cap H \) or a part of it.

**Proposition 3.11.** Consider a polyblock \( P \) with a vertex set \( \mathcal{T} \). Let \( \mathcal{T}' \) be the set that is obtained from \( \mathcal{T} \) by deleting all \( \{ v \in \mathcal{T} | v \notin \mathcal{H} \} \) and all \( \{ v \in \mathcal{T} | [0, v] \cap G \cap H = \emptyset \} \), replacing every remaining \( v \in \mathcal{T} \) by \( v' \) that satisfies

\[
v'_i = \max \{ x_i | x \in G \cap H \cap [0, v] \} \quad \forall i = 1, \ldots, n.
\]

and finally removing all improper vertices. The resulting \( \mathcal{T}' \) generates a polyblock \( P' \) such that

\[
G \cap H \cap P' \subset P' \subset P.
\]

The procedure in Proposition 3.11 is illustrated in Figure 3.5. Here, vertices \( v_1 \) and \( v_5 \) were removed because they do not belong to \( \mathcal{H} \). \( v_2, v_3, \) and \( v_4 \) are “tightened” to \( v'_2, v'_3, \) and \( v'_4 \), while \( v'_4 \) can further be
Fig. 3.4 Tightening a box.

Fig. 3.5 Tightening a polyblock.
removed since it is not proper. With this, Lines 11 and 13 of Algorithm 2 can be refined as

11: Remove from $T_{k+1}$ the improper vertices and all vertices such that $\{v \in T_{k+1} | v \notin H \text{ or } f(v) \leq CBV_k + \epsilon\}$. If $T_{k+1} \neq \emptyset$, apply the tightening procedure in Proposition 3.11.

13: until $T_{k+1} = \emptyset$.

For the convenience of readers, the modified polyblock outer approximation is presented in Algorithm 3 with enhancements discussed in this section.
Algorithm 3 Enhanced Polyblock Outer Approximation Algorithm

**Input:** An upper semicontinuous increasing function $f(\cdot) : \mathcal{R}_+^n \to \mathcal{R}$, a compact normal set $G \subset \mathcal{R}_+^n$, and a closed conormal set $H \subset \mathcal{R}_+^n$ such that $G \cap H \neq \emptyset$

**Output:** an $\epsilon$-optimal solution $x^*$

1. **Initialization:** Let the initial polyblock $P_1$ be box $[0,b]$ that encloses $G \cap H$. The vertex set $T_1 = \{b\}$. Let $\epsilon \geq 0$ be a small positive number. $CBV_0 = -\infty$. $k = 0$.

2. **repeat**

3. $k = k + 1$.

4. From $T_k$, select $z_k \in \text{argmax}\{f(z) | z \in T_k\}$.

5. Compute $\pi_G(z_k)$, the projection of $z_k$ on the upper boundary of $G$.

6. **if** $\pi_G(z_k) = z_k$, i.e., $z_k \in G$ **then**

7. $\bar{x}_k = z_k$ and $CBV_k = f(z_k)$.

8. **else**

9. If $\pi_G(z_k) \in G \cap H$ and $f(\pi_G(z_k)) \geq CBV_{k-1}$, then let the current best solution $\bar{x}_k = \pi_G(z_k)$ and $CBV_k = f(\pi_G(z_k))$. Otherwise, $\bar{x}_k = \bar{x}_{k-1}$ and $CBV_k = CBV_{k-1}$.

10. Let $x = \pi_G(z_k)$ and

    $\mathcal{T}_{k+1} = (\mathcal{T}_k \setminus \mathcal{T}_*) \cup \{v^i = v + (x_i - v_i)e^i | v \in \mathcal{T}_*, i \in \{1, \ldots, n\}\}$,

    where $\mathcal{T}_* = \{v \in \mathcal{T}_k | v > x\}$.

11. **Remove from $\mathcal{T}_{k+1}$ the improper vertices and all vertices such that $\{v \in \mathcal{T}_{k+1} | v \notin H$ or $f(v) \leq CBV_{k+1} + \epsilon\}$. If $\mathcal{T}_{k+1} \neq \emptyset$, apply the tightening procedure in Proposition 3.11.

12. **end if**

13. **until** $\mathcal{T}_{k+1} = \emptyset$.

14. Let $x^* = \bar{x}_k$ and terminate the algorithm.
3.4 Discrete Monotonic Optimization

In many applications, the variables (or some of the variables) to be optimized are confined to a finite set. For example, some entries of the variable vector $\mathbf{x}$ may be subject to Boolean constraints like $x_i \in \{0, 1\}, \ i = 1, \ldots, n$. In general, we say $x_i$ is confined to a finite set $S_i$, such that the vector $[x_1, \ldots, x_n] \in \mathcal{S} = S_1 \times \cdots \times S_n$. As such, the canonical form of discrete monotonic optimization problems is written as

$$\max \{ f(\mathbf{x}) | \mathbf{x} \in \mathcal{G} \cap \mathcal{H} \cap \mathcal{S} \},$$  \hspace{1cm} (3.4)

where $\mathcal{G}$ and $\mathcal{H}$ are defined as before.

In the rest of this section, we extend the polyblock outer approximation algorithm for the continuous monotonic optimization to obtain an algorithm that solves the discrete Problem (3.4). Note that the continuous algorithm only achieves an $\epsilon$-optimal algorithm in finite steps. The exact optimal solution can be obtained only through infinite iterations. The discrete algorithm, however, can compute an exact optimal solution in finitely many steps.

Let us first introduce the lower $S$-adjustment operation.

Definition 3.1 (Lower $S$-adjustment). Given any point $\mathbf{x} \in [0, b]$, we write the lower $S$-adjustment of $\mathbf{x}$ as $[\mathbf{x}]_S = \tilde{\mathbf{x}}$, where the point $\tilde{\mathbf{x}}$ satisfies

$$\tilde{x}_i = \max \{ \xi | \xi \in S_i \cup \{0\}, \xi \leq x_i \} \ \forall \ i = 1, \ldots, n$$  \hspace{1cm} (3.5)

The polyblock outer approximation algorithm can be easily extended to solve the discrete problem through the above-defined lower $S$-adjustment operation using the following propositions.

Proposition 3.12. Let $\mathcal{P}$ be a polyblock that encloses the feasible set $\mathcal{G} \cap \mathcal{H} \cap \mathcal{S}$ and $\mathbf{x} \in \partial^+ \mathcal{G}$. Then, $\mathcal{P} \setminus \mathcal{K}_x^+$ still encloses $\mathcal{G} \cap \mathcal{H} \cap \mathcal{S}$. Moreover, suppose that $\tilde{\mathbf{x}} = [\mathbf{x}]_S$ is the lower $S$-adjustment of $\mathbf{x}$. Then, $\mathcal{P} \setminus \mathcal{K}_{\tilde{\mathbf{x}}}^+$ also encloses $\mathcal{G} \cap \mathcal{H} \cap \mathcal{S}$. 
The first half of Proposition 3.12 is straightforward from the separation property of normal sets, in the sense that cutting off a cone $K^+\mathbf{x}$ where $\mathbf{x} \in \partial^\dagger \mathcal{G}$ will not exclude any points in $\mathcal{G}$. The second half of the proposition is due to the fact that the cone $K^+_{\mathbf{x}}$ does not include more points in $S$ than $K^+_{\mathbf{x}}$.

**Proposition 3.13.** Let $\mathcal{P} \supset \mathcal{G} \cap \mathcal{H} \cap S$ be an enclosing polyblock with vertex set $\mathcal{T} \subset \mathbb{R}^n_+$. Then, another polyblock $\mathcal{P}'$ with vertex set $\mathcal{T}' = \{ \tilde{\mathbf{v}} | \tilde{\mathbf{v}} = \lfloor \mathbf{v} \rfloor_S^*, \mathbf{v} \in \mathcal{T} \}$ also encloses $\mathcal{G} \cap \mathcal{H} \cap S$.

Proposition 3.13 can be verified by noting that box $[0, \mathbf{v}]$ encloses as many points in $S$ as box $[0, \lfloor \mathbf{v} \rfloor_S^*]$.

Similar to the case with continuous monotonic optimization, the polyblock outer approximation for discrete monotonic optimization generates a nested sequence of polyblocks outer approximating the feasible set:

$$\mathcal{P}_1 \supset \mathcal{P}_2 \supset \cdots \supset \mathcal{P}_k \supset \cdots \supset \mathcal{G} \cap \mathcal{H} \cap S.$$ 

We can apply an enhancement procedure similar to the one in Subsection 3.3.1 to remove suboptimal vertices of the polyblocks, i.e., the ones with function values smaller than the current best value. Then, the sequence of polyblocks satisfies

$$\mathcal{P}_1 \supset \mathcal{P}_2 \supset \cdots \supset \mathcal{P}_k \supset \cdots \supset \mathcal{G} \cap \mathcal{H} \cap S_{(k)},$$

where $S_{(k)} = \{ \mathbf{x} \in S | f(\mathbf{x}) > \text{CBV}_{k-1} \}$, where CBV$_{k-1}$ is the current best value known from the last round.

The algorithm for discrete monotonic optimization differs from the one for continuous monotonic optimization mainly in the following two building blocks.

1. **Computing the upper boundary point from $\mathbf{z} \in \mathbb{R}^n_+ \setminus \mathcal{G}$**

The procedures in Subsection 3.2.1 for continuous monotonic optimization step can be slightly adjusted for discrete problems as follows by the property of lower $S$-adjustment in Proposition 3.12.
3.4 Discrete Monotonic Optimization

Input: $z_k$, $G$, $S_{(k)}$

Output: $\tilde{\pi}_G(z_k)$ such that $\tilde{\pi}_G(z_k) \in G \cap S_{(k)}$ and $K^+_{\tilde{\pi}_G(z_k)} \cap (G \cap H \cap S_{(k)}) = \emptyset$.

1: Calculate $\pi_G(z_k)$ according to Algorithm 1.
2: If $\pi_G(z_k) \in S_{(k)}$, then let $\tilde{\pi}_G(z_k) = \pi_G(z_k)$.
3: Otherwise, if $\pi_G(z_k) \notin S^*_{(k)}$, then let $\tilde{\pi}_G(z_k) = \lfloor \pi_G(z_k) \rfloor_{S_{(k)}}$.

(2) Generating the new polyblock

This procedure of generating new enclosing polyblock is largely the same as in the continuous case, except that the vertices of the new enclosing polyblock must be lower $S$-adjusted. The detailed steps are given as follows.

Input: The vertex set $T_k$ of $P_k$, $\tilde{\pi}_G(z_k)$, $CBV_k$.

Output: A proper vertex set $T_{k+1}$ of $P_{k+1}$, such that $P_k \supset P_{k+1} \supset G \cap H \cap S_{(k+1)}$ and $T_{k+1} \subset (H \cap S_{(k+1)})$.

1: Let $x = \tilde{\pi}_G(z_k)$ and

\[ T = (T_k \setminus T_o) \cup \{v^i = v + (x_i - v_i)e^i | v \in T_o, i \in \{1, \ldots, n\}\}, \]

where $T_o = \{v \in T_k | v > x\}$.

2: Let $S_{(k+1)} = \{x \in S | f(x) > CBV_k\}$.
3: $T_{k+1} = \{\tilde{v} = \lfloor v \rfloor_{S_{(k+1)}} | f(\tilde{v}) > CBV_k, v \in T\}$.
4: Remove from $T_{k+1}$ improper vertices and vertices $\{v \in T_{k+1} | v \notin H\}$.

We are now ready to summarize the discrete polyblock outer approximation algorithm in Algorithm 4.

It can be proved that Algorithm 4 converges to the optimal solution within a finite number of steps. In a more general case where only some of the entries in $x$ are discrete while others are continuous, one can show that the algorithm converges to the optimal solution in finite number of steps.
Algorithm 4 Polyblock Outer Approximation Algorithm for Discrete Monotonic Optimization

**Input:** A upper semicontinuous increasing function \(f(\cdot) : \mathbb{R}_+^n \rightarrow \mathbb{R}\), a compact normal set \(G \subset \mathbb{R}_+^n\), a closed conormal set \(H \subset \mathbb{R}_+^n\) such that \(G \cap H \neq \emptyset\), and a discrete set \(S\)

**Output:** an optimal solution \(x^*\)

1: Initialization: Let \([0, b]\) be a box that encloses \(G \cap H\). Let \(p_1 = [0, \tilde{b}]\), where \(\tilde{b}\) is the lower \(S\)-adjustment of \(b\). The vertex set \(T_1 = \{\tilde{b}\}\). Let \(\text{CBV}_0 = -\infty. k = 0\).

2: repeat
3: \(k = k + 1\).
4: From \(T_k\), select \(z_k \in \text{argmax}\{f(z)|z \in T_k\}\).
5: if \(z_k \in G \cap H \cap S\) then
6: \(\bar{x}_k = z_k, T_{k+1} = \emptyset\).
7: else
8: Compute \(\pi_G(z_k)\), the projection of \(z_k\) on the upper boundary of \(G\).
9: If \(\pi_G(z_k) \in S(\bar{k})\), then let \(\bar{\pi}_G(z_k) = \pi_G(z_k)\). Otherwise, \(\bar{\pi}_G(z_k) = \lfloor \pi_G(z_k) \rfloor_{S(\bar{k})}\).
10: If \(\bar{\pi}_G(z_k) \in G \cap H\) and \(f(\bar{\pi}_G(z_k)) \geq \text{CBV}_{k-1}\), then let the current best solution \(\bar{x}_k = \bar{\pi}_G(z_k)\) and \(\text{CBV}_k = f(\bar{\pi}_G(z_k))\).
11: Let \(x = \bar{\pi}_G(z_k)\) and \(\mathcal{T} = (T_k \setminus T_{k+1}) \cup \{v^i = v + (x_i - v_i)e_i|v \in T, i \in \{1, \ldots, n\}\}\), where \(T_k = \{v \in T_k|v > x\}\).
12: Let \(S_{(k+1)} = \{x \in S|f(x) > \text{CBV}_k\}\).
13: \(T_{k+1} = \{\bar{v} = [v]_{S_{(k+1)}}|f(v) > \text{CBV}_k, v \in T\}\).
14: Remove from \(T_{k+1}\) improper vertices and vertices \(\{v \in T_{k+1}|v \notin H\}\).
15: end if
16: until \(T_{k+1} = \emptyset\).
17: Output \(x^* = \bar{x}_k\) as the optimal solution.
Part II

Applications
In this part, we illustrate the power of monotonic optimization through four sample applications. Section 4 discusses the nonconvex power control in wireless interference channels. This problem can be formulated into a GLFP, which is a special case of monotonic optimization problems. Through this example, we also demonstrate the trick of “shift of origin” to guarantee the convergence of the algorithm. Section 5 discusses power controlled scheduling problems, where we reduce the number of variables by exploiting the convexity of some set. Section 6 extends the discussion of Section 4 to multi-antenna systems, where we optimize the transmit beamforming in MISO interference channels. Here, we illustrate how to deal with vector variables in monotonic optimizations. Finally, Section 7 concerns the network utility maximization in random access networks, where we need to maximize an increasing function of polynomials. Through this problem, we illustrate the use of auxiliary variables to convert a “difference of monotonic” optimization problem to a canonical monotonic optimization problem.
The material in this section is mainly based on [24].

4.1 System Model and Problem Formulation

We consider a wireless network with a set \( \mathcal{L} = \{1, \ldots, n\} \) of distinct links. Each link includes a transmitter node \( T_i \) and a receiver node \( R_i \). The channel gain between node \( T_i \) and node \( R_j \) is denoted by \( G_{ij} \), which is determined by various factors such as path loss, shadowing, and fading effects. The complete channel matrix is denoted by \( \mathbf{G} = [G_{ij}] \).

Let \( p_i \) denote the transmission power of link \( i \) (i.e., from node \( T_i \) to node \( R_i \)), and \( \eta_i \) denote the received noise power on link \( i \) (i.e., measured at node \( R_i \)). The received signal to interference-plus-noise ratio (SINR) of link \( i \) is

\[
\gamma_i(p) = \frac{G_{ii}p_i}{\sum_{j \neq i} G_{ji}p_j + \eta_i},
\]  

\( (4.1) \)

\(^1\)For example, this could represent a network snapshot under a particular schedule of transmissions determined by an underlying routing and MAC protocol.
and the data rate calculated based on the Shannon capacity formula is
\[ \log_2(1 + \gamma_i(p)) \] To simplify notations, we use \( p = (p_i, \forall i \in L) \), \( P_{\text{max}} = (P_{\text{max}}^i, \forall i \in L) \), and \( \gamma(p) = (\gamma_i(p), \forall i \in L) \) to represent the transmission power vector, the maximum transmission power vector, and achieved SINR vector of all links, respectively.

We want to find the optimal power allocation \( p^* \) that maximizes a system utility subject to individual data rate constraints. Mathematically, we aim to solve the following optimization problem:

\[
\begin{align*}
\max_{p} & \quad U(\gamma(p)) \\
\text{subject to} & \quad \gamma_i(p) \geq \gamma_{i,\text{min}}, \forall i = 1, \ldots, n, \\
& \quad 0 \leq p_i \leq P_{\text{max}}^i, \forall i = 1, \ldots, n.
\end{align*}
\]

Here, \( U(\cdot) \) is the system utility and is an increasing function of \( \gamma \). \( \gamma_{i,\text{min}} > 0 \) is the minimum SINR requirement of link \( i \). In most cases, the system utility \( U(\cdot) \) is a summation of users’ individual utilities, i.e., \( U(\gamma(p)) = \sum_i U_i(\gamma_i(p)) \). For example, we will maximize the total system throughput if \( U_i(\gamma_i(p)) = \log_2(1 + \gamma_i(p)) \), the proportional fairness if \( U_i(\gamma_i(p)) = \log(\log_2(1 + \gamma_i(p))) \), and the max–min fairness if \( U(\gamma(p)) = \min_i \gamma_i(p) \). Note that we do not assume any concavity or differentiability of \( U(\cdot) \). The polyblock outer approximation algorithm works as long as \( U(\cdot) \) is monotonically increasing.

Problem (4.2) is not in the canonical form (Equation (2.1)) of monotonic optimization, in that the objective function is not an increasing function of the variable \( p \). One may notice that the problem has a hidden monotonicity structure, in the sense that the objective function is an increasing function of a positive-valued function \( \gamma(p) \). Indeed, Equation (4.2) is a GLFP defined in Definition 2.9 which can be transformed into the canonical form. More specifically, Problem (4.2) is equivalent to

\[
\begin{align*}
\max_{y} & \quad U(y) \\
\text{subject to:} & \quad y \in \mathcal{G} \cap \mathcal{H},
\end{align*}
\]

\[ To better model the achievable rates in a practical system, we can re-normalize \( \gamma_i \) by \( \beta \gamma_i \), where \( \beta \in [0, 1] \) represents the system’s “gap” from capacity. Such modification, however, does not change the analysis in this section.
where $G = \{ y | 0 \leq y_i \leq \gamma_i(p), \forall i \in L, 0 \leq p \leq P_{\text{max}} \}$ and $H = \{ y | y_i \geq \gamma_{i,\text{min}}, \forall i \in L \}$.

The optimal solution to Problem (4.2) can be recovered from that to Equation (4.3), denoted by $y^*$, through solving $n$ linear equations $G_{ii}p_i = y^*_i (\sum_{j \neq i} G_{ji}p_j + \eta_i)$ for each link $i$.

### 4.2 Algorithm

Formulated in the canonical form, Problem (4.3) can be solved with Algorithm 3 as long as $\gamma_{\text{min}}$ is feasible. In the following, we first discuss how to efficiently check the feasibility of $\gamma_{\text{min}}$. Then, we will elaborate the execution of two key steps, i.e., initialization (Line 1 of Algorithm 3) and projection (Line 5 of Algorithm 3), for the particular problem of power control.

#### 4.2.1 Feasibility Check

In this subsection, we discuss the feasibility of $\gamma_{\text{min}}$ when the transmit power $p_i$ is constrained by Equation (4.2c). Consider the following matrix $B$

$$B_{ij} = \begin{cases} 0, & i = j, \\ \frac{\gamma_{\text{min}}G_{ji}}{G_{ii}}, & i \neq j. \end{cases}$$

The feasibility of $\gamma_{\text{min}}$ can be checked by Lemma 4.1 thanks to the Perron Frobenius theorem.

**Lemma 4.1.** There exists a power vector $p \geq 0$ that satisfies $\gamma(p) \geq \gamma_{\text{min}}$ if and only if $\rho(B) < 1$, where $\rho(\cdot)$ denotes the maximum eigenvalue of the matrix. Furthermore, the nonnegative power vector $p$ that satisfies $\gamma(p) = \gamma_{\text{min}}$ can be calculated as follows if $\rho(B) < 1$,

$$p = (I - B)^{-1}u,$$

where $I$ is an $n \times n$ identity matrix and $u$ is an $n \times 1$ vector with elements

$$u_i = \frac{\gamma_{i,\text{min}}\eta_i}{G_{ii}}.$$
If the power vector calculated in Equation (4.5) also satisfies the maximum transmit power constraint \( p \leq P_{\text{max}} \), then \( \gamma_{\text{min}} \) is achievable by a power vector \( p \in [0, P_{\text{max}}] \).

For easy reading, we rewrite the procedure in Lemma 4.1 in Algorithm 5.

**Algorithm 5 Checking the Feasibility of \( \gamma_{\text{min}} \)**

**Input:** \( \gamma_{\text{min}}, P_{\text{max}} \).

**Output:** An indicator whether \( \gamma_{\text{min}} \) can be achieved by transmit power \( p \in [0, P_{\text{max}}] \).

1. Generate \( B \) and \( u \) according to Equations (4.4) and (4.6), respectively.
2. If \( \rho(B) \geq 1 \) then
   3. Output: \( \gamma_{\text{min}} \) is infeasible.
3. Else
   4. Compute \( p = (I - B)^{-1}u \).
5. If \( p \leq P_{\text{max}} \) then
   6. Output: \( \gamma_{\text{min}} \) is feasible.
6. Else
   7. Output: \( \gamma_{\text{min}} \) is infeasible.
11. End if

4.2.2 Initialization and Computing the Upper Boundary Point

To initialize, we find a vertex \( b \) such that box \([0, b]\) contains the normal hull of the entire feasible SINR region. One simple way is to let the \( i^{th} \) entry \( b_i \) be the upper bound of the maximum achievable SINR of the \( i^{th} \) link, i.e., \( b_i = \max_p \gamma_i(p) = \frac{G_{ii}P_{\text{max}}}{\eta} \). One can also set a tighter initialization at the cost of some additional computation.

To calculate the upper boundary projection \( \pi_G(z_k) \), we resort to the bisection search method in Algorithm 1. Here, the main complexity is to check whether \( \gamma = \bar{\alpha}z_k \) is in \( \mathcal{G} \), or in other words, whether \( \gamma = \bar{\alpha}z_k \) is
achievable by a power vector $\mathbf{p}$ that lies in $[0, \mathbf{P}^{\text{max}}]$. Luckily, the complexity of feasibility check for the power control problem is quite low, as discussed in the last subsection. We can easily check the feasibility of $\gamma = \hat{\alpha} \mathbf{z}$ using Algorithm 5 in each iteration of Algorithm 1.

Knowing how to carry out these two steps, the execution of other steps of Algorithm 3 is straightforward.

### 4.2.3 Shift of Origin

Recall from Subsection 3.2.4 that a key condition for the polyblock outer approximation algorithm to converge is that $\mathcal{G} \cap \mathcal{H} \subset \mathbb{R}^{n}_{++}$. This condition holds in Problem (4.3) if $\gamma_{i, \min} > 0$ for all $i$. In case when some $\gamma_{i, \min}$’s are zero, we can use the trick of “shift of origin” to recover the convergence.

Define $\hat{\gamma}(\mathbf{p}) = \gamma(\mathbf{p}) + 1$ and write Problem (4.2) as

$$\max_{\mathbf{p}} \quad U(\hat{\gamma}(\mathbf{p}) - 1)$$

subject to $\hat{\gamma}_i(\mathbf{p}) \geq 1 + \gamma_{i, \min}, \, \forall i = 1, \ldots, n,$

$$0 \leq p_i \leq P_i^{\text{max}}, \, \forall i = 1, \ldots, n,$$

which can then be transformed to

$$\max_{\mathbf{y}} \quad U(\mathbf{y} - 1)$$

subject to: $\mathbf{y} \in \mathcal{G} \cap \mathcal{H},$

where $\mathcal{G} = \{\mathbf{y} | 0 \leq y_i \leq \gamma'_i(\mathbf{p}), \forall i \in \mathcal{L}, 0 \leq \mathbf{p} \leq \mathbf{P}^{\text{max}}\}$ and $\mathcal{H} = \{\mathbf{y} | y_i \geq 1 + \gamma_{i, \min}, \forall i \in \mathcal{L}\}$. Now, since $1 + \gamma_{i, \min} > 0$, we can guarantee that $\mathcal{G} \cap \mathcal{H} \subset \mathbb{R}^n_{++}$. The same technique will also be used in the next section to solve the power controlled scheduling problem.

### 4.3 Numerical Results

The globally optimal power control algorithm introduced above is referred to as the MAPEL algorithm in [24]. One key application of MAPEL is that it enables us to easily compute and understand the characteristics of the global optimal power control solutions for an arbitrary wireless network. This will provide hints for designing
good heuristic and practical algorithms, and will provide a benchmark for evaluating the performance of these low-complexity heuristic algorithms. With MAPEL, we are able to give quantitative measurements of the algorithms’ performance, such as the probability of achieving global optimal solution and the suboptimality gap, under a wide range of network scenarios.

We first use MAPEL to find the global optimal power allocation that maximizes the total network throughout, i.e., \( U_i(\gamma_i(p)) = \log_2(1 + \gamma_i(p)) \). For simplicity, we assume that \( \gamma_{i,\text{min}} = 0 \) for each link. Consider a 4-link network in Figure 4.1 as a simple illustrative example. All links have the same length of 4 m and the same priority weight. The distances between \( T_i \) to \( R_j \) for \( i \neq j \), denoted by \( l_{ij} \), are parameterized by \( d \). The four links have different direct channel gains: \( G_{11} = 1 \), \( G_{22} = 0.75 \), \( G_{33} = 0.50 \), and \( G_{44} = 0.25 \). The cross-channel gains are \( G_{ij} = l_{ij}^{-1} \), the maximum transmission power \( P^{\text{max}} = (0.7 0.8 0.9 1.0) \) mW, and background noise \( n_i = 0.1 \) \( \mu \)W for all links \( i \). In Figure 4.2, the optimal transmission power of each link is plotted against the topology parameter \( d \). We can see when the links are very close to each other (i.e., \( d < 2.5 \) m), only the link with the largest channel gain (i.e., Link 1) is active with the maximum transmission power \( P_1^{\text{max}} \), while all the other links remain silent. When \( d \) increases, the transmission power of Link 2 jumps from 0 to \( P_2^{\text{max}} \) [3] As \( d \) further increases, Link 3 starts

\[ \text{Fig. 4.1 The network topology of four links.} \]

[3] In fact, if there are only two active links, they must both transmit at the maximum power.
to transmit, followed by Link 4. In this symmetric example, we notice that the global optimal solution activates the links according to their direct channel gains. When the network graph is asymmetric, the global optimal solution can be more complicated.

We now proceed to show how MAPEL can be used to benchmark other heuristic power control algorithms that also aim to maximize the network throughput. Here, we will review three representative heuristic algorithms, namely the Geometric Programming (GP) algorithm [4], the Signomial Programming Condensation (SPC) Algorithm [4], and the Asynchronous Distributed Pricing (ADP) Algorithm [11]. Such a comparison can be used for any future heuristic algorithms that are designed to solve the same power control problem.

In Figure 4.3, we compare the average system throughput obtained by MAPEL, GP, SPC, and ADP algorithms under different network densities. For each given number of links $n$, we place the links randomly in a 10 m-by-10 m area, and average the results over 500 random link placements. The length of each link is uniformly distributed within [1 m, 2 m]. The maximum transmission power $P_{\text{max}} = 1 \text{ mW}$, background noise $n_i = 0.1 \mu\text{W}$, and the initial power allocation for SPC and ADP is fixed at $P_{\text{max}}/2$ for each link. We vary the total number of links $n$ from 1 to 10. The results show that the performance of the

Fig. 4.2 The relationship between optimal transmission power and distance $d$.
SPC is usually within 98% of the global optimal benchmark achieved by MAPEL, while the ADP algorithm can achieve an average performance of 90% of the optimal. Moreover, the GP algorithm works reasonably well when the network density is low, where all (or most) links are active and some of them are in the high SINR regime (which is the assumption of the GP algorithm). However, suboptimality gap of the GP algorithm becomes much larger when the network becomes denser, where many links need to be silent in order to avoid heavy interferences to their neighbors.
The material in this section is mainly based on [23].

In the last section, the transmit power for each link does not vary with time once it is determined. This may result in a small achievable rate region, especially in dense networks. Take a two-link network in Figure 5.1 as an example, where the two links are close to each other. Suppose that $P_{\text{max}} = 1.0$ W for both links, and the noise power at each receiver is $10^{-4}$ W. With power control, the achievable rate region is given in Figure 5.2(a), where each point in the rate region corresponds to a transmit power vector $p \in [0, P_{\text{max}}]$. In particular, the maximum sum rate $10.97$ bps/Hz is achieved by activating Link 2 only, i.e. $p_1 = 0$ and $p_2 = 1.0$ W. If a minimum rate requirement of 1 bps/Hz is imposed on both links, then the maximum sum rate drastically drops to $3.9$ bps/Hz (where $r_1 = 1$ bps/Hz and $r_2 = 2.9$ bps/Hz), which can be achieved by $p_1 = 1$ W and $p_2 = 0.998$ W.
On the other hand, if we allow time sharing of different transmit power vectors, then the rate region can be significantly expanded, as illustrated in Figure 5.2(b). In particular, a sum rate that can be achieved is increased to 10.87 bps/Hz (where \( r_1 = 1 \) bps/Hz and \( r_2 = 9.87 \) bps/Hz) when the minimum data rate requirement is 1 bps/Hz for both links. This is achieved by time sharing two power vectors \( \mathbf{p} = [1, 0] \) W and \( \mathbf{p} = [0, 1] \) W.

In this section, we focus on the scheduling of power vectors, or power controlled scheduling, i.e., how much time to be allocated to which transmit power vector.
5.1 System Model and Problem Formulation

We consider the same $n$-link network model as in Section 4. The power controlled scheduling problem can be formulated as follows

$$\max_{p(t)} U(r) \quad (5.1a)$$

subject to

$$r_i = \frac{1}{T} \int_0^T \log_2(1 + \gamma_i(p(t))) dt \geq r_i^{\min}, \quad \forall i = 1, \ldots, n \quad (5.1b)$$

$$0 \leq p(t) \leq P^{\max}, \forall t \in [0, T], \quad (5.1c)$$

where $U(\cdot)$ is an increasing function that is not necessarily concave or differentiable. $r = (r_i, i \in \mathcal{L})$ denotes the average data rate of users. $r_i^{\min} > 0$ denotes the minimum data rate requirement of each user. In this formulation, scheduling has been integrated into the time-varying power allocation $p(t)$. For easy discussion, we assume that $U(r)$ is additive, i.e., $U(r) = \sum_i U_i(r_i)$, in the following. The work, however, can be easily extended to nonadditive $U$’s.

5.1.1 Variable Discretization

Problem (5.1) is more challenging than Problem (4.2), mainly because the problem size of Equation (5.1) is infinite as $t$ is continuous during $[0, T]$. Here, we address the issue by Lemma 5.1 and Theorem 5.2. Before presenting the Lemma and Theorem, we first define achievable instantaneous data rate set $\mathcal{R}(t)$ and the achievable average data rate set $\bar{\mathcal{R}}$:

$$\mathcal{R}(t) = \{ r(t) | r_i(t) = \log_2(1 + \gamma_i(p(t))), \; 0 \leq p(t) \leq P^{\max}, \forall i \in \mathcal{L} \}, \quad (5.2)$$

and

$$\bar{\mathcal{R}} = \left\{ r | r_i = \frac{1}{T} \int_0^T \log_2(1 + \gamma_i(p(t))) dt, \; 0 \leq p(t) \leq P^{\max}, \quad \forall i \in \mathcal{L}, \forall t \in [0, T] \right\}. \quad (5.3)$$

The instantaneous data rate set $\mathcal{R}(t)$ is the set of all data rates achievable by power allocation at time instant $t$. On the other hand, the
average data rate set $\bar{R}$ is the set of all achievable \textit{average} data rates during a scheduling period $T$ through time-varying power allocation.

\textbf{Remark 5.1.} Under the assumption that channels conditions remain unchanged during time period $[0, T]$, $\mathcal{R}(t)$ is the same for all $t \in [0, T]$.

By the standard convexity argument, Remark 5.1 leads to Lemma 5.1 [3].

\textbf{Lemma 5.1.} The achievable average data rate set $\bar{R}$ is the convex hull of the instantaneous data rate set $\mathcal{R}(t)$, i.e., $\bar{R} = \text{Convex Hull}\{\mathcal{R}(t)\}$.

\textbf{Theorem 5.2.} By Caratheodory theorem [3] and Lemma 5.1, the number of elements in $\mathcal{R}(t)$ that is needed to construct an arbitrary average data rate vector $\mathbf{r} = (r_i, \forall i \in \mathcal{L})$ in $\bar{R}$ is no more than $n + 1$.

Theorem 5.2 implies that an arbitrary average data rate vector $\mathbf{r}$ can be achieved by dividing $[0, T]$ into $n + 1$ intervals with lengths $\beta^1, \ldots, \beta^{n+1}$ and assigning power vectors $\mathbf{p}^1, \ldots, \mathbf{p}^{n+1}$ to these intervals. Therefore, the constraints of Problem 5.1 can be replaced by

\begin{align}
    r_i &= \sum_{k=1}^{n+1} \beta^k \log_2(1 + \gamma_i(\mathbf{p}^k)) \geq r_i^{\min}, \forall i \in \mathcal{L}, \quad (5.4a) \\
    \sum_{k=1}^{n+1} \beta^k &= 1, \beta^k \geq 0, \forall k \in \mathcal{K} = \{1, \ldots, n + 1\}, \quad (5.4b) \\
    0 &\leq \mathbf{p}^k \leq \mathbf{P}^{\max}, \forall k \in \mathcal{K} = \{1, 2, \ldots, n + 1\}, \quad (5.4c)
\end{align}

where we have normalized $\beta^k$ with respect to $T$. By doing so, we have turned an infinite number of variables $\mathbf{p}(t)$ to a finite number of variables $\mathbf{p}^k$ without compromising the optimality of the problem. The power controlled scheduling problem is now equivalent to finding a piecewise constant power allocation that has $n + 1$ degrees of freedom in the time domain.
5.1.2 Feasible Set Simplification

One can use similar tricks as in the last section to convert the joint scheduling and power control problem into a canonical monotonic optimization problem. However, a close observation of Equation (5.4) indicates that the conormal set $H$, which results from the first constraint, could be quite complicated. To resolve the issue, let us define $\hat{U}_i(r_i)$ to be

$$
\hat{U}_i(r_i) = \begin{cases} 
U_i(r_i) & \text{if } r_i \geq r_i^{\min}, \\
-\infty & \text{otherwise.} 
\end{cases}
$$

(5.5)

Similar to $U_i(r_i)$, $\hat{U}_i(r_i)$ is a monotonic increasing function. With Equation (5.5), Problem (5.1) can be rewritten as

$$
\max_{\beta, p_k} \sum_{i=1}^n \hat{U}_i \left( \sum_{k=1}^{n+1} \beta_k \log_2(1 + \gamma_i(p_k)) \right)
$$

subject to

$$
\sum_{k=1}^{n+1} \beta_k \leq 1, \beta \geq 0 
$$

(5.6b)

$$
0 \leq p_k \leq P_{\max}, \forall k \in K.
$$

(5.6c)

Note that Problem (5.6) is equivalent to Problem (5.1) when Problem (5.1) is feasible. On the other hand, the objective function of Problem (5.6) is equal to $-\infty$ if and only if Problem (5.1) is infeasible.

5.1.3 The Canonical Form

Now we are ready to reformulate Problem (5.6) into a canonical form monotonic programming problem.

Let $(\beta, y)$ denote the concatenation of two vectors $\beta = (\beta_1, \ldots, \beta_{n+1})$ and $y = (y_1^1, \ldots, y_1^n, \ldots, y_{n+1}^1, \ldots, y_{n+1}^n)$. Since the function $\hat{U}_i(r_i)$ is non-decreasing in $r_i$, it is easy to see that the function $\Phi((\beta, y)) = \sum_{i=1}^n \hat{U}_i \left( \sum_{k=1}^{n+1} \beta_k \log_2(1 + y_i^k) \right)$ is a nondecreasing function on $\mathbb{R}_{+}^{(n^2 + 2n+1)}$. That is, for any two vectors $(\beta_1, y_1)$ and $(\beta_2, y_2)$ such that $(\beta_1, y_1) \geq (\beta_2, y_2)$, we have $\Phi((\beta_1, y_1)) \geq \Phi((\beta_2, y_2))$. We further note that $\gamma_i(p^k)$ for all $i$ and $k$ is nonnegative, and the time
fraction vector $\beta$ is nonnegative as well. Based on these observations, Problem (5.6) can be rewritten into the following canonical form:

$$\max_{(\beta, y)} \Phi((\beta, y)) = \sum_{i=1}^{n} \hat{U}_i (\sum_{k=1}^{n+1} \beta^k \log_2 (1 + y^k_i))$$

subject to $$(\beta, y) \in G,$$

where the feasible set

$$G = \left\{ (\beta, y) \mid \sum_{k=1}^{n+1} \beta^k \leq 1, \ 0 \leq y^k_i \leq \gamma_i(p^k), \ \forall i \in \mathcal{L}, \forall k \in \mathcal{K}, (\beta, p) \in \mathcal{P}_\beta \right\}$$

with

$$\mathcal{P}_\beta = \left\{ (\beta, p) \mid \beta^k \geq 0 \text{ and } 0 \leq p^k_i \leq P^\max_i, \ \forall i \in \mathcal{L}, \forall k \in \mathcal{K} \right\}.$$ (5.9)

### 5.1.4 Shift of Origin

Note that the feasible set here is not strictly bounded away from 0. Similar as before, the issue can be addressed by the trick of “shift of origin”. Define $\tilde{\beta}^k = \beta^k + 1$ and $\tilde{\gamma}_i(p^k) = \gamma_i(p^k) + 1$. Problem (5.6) becomes

$$\max_{\tilde{\beta}, p} \sum_{i=1}^{n} \hat{U}_i \left( \sum_{k=1}^{n+1} (\tilde{\beta}^k - 1) \log_2 (\tilde{\gamma}_i(p^k)) \right)$$

subject to $$(\tilde{\beta}, p) \in \mathcal{G} \cap \mathcal{H},$$

which has an equivalent canonical form of

$$\max_{(\tilde{\beta}, y)} \Phi((\tilde{\beta}, y)) = \sum_{i=1}^{n} \hat{U}_i \left( \sum_{k=1}^{n+1} (\tilde{\beta}^k - 1) \log_2 (y^k_i) \right)$$

subject to $$(\tilde{\beta}, y) \in \mathcal{G} \cap \mathcal{H},$$

where the feasible set

$$\mathcal{G} = \left\{ (\tilde{\beta}, y) \mid \sum_{k=1}^{n+1} (\tilde{\beta}^k - 1) \leq 1, \ 0 \leq y^k_i \leq \tilde{\gamma}_i(p^k), \ \forall i \in \mathcal{L}, \forall k \in \mathcal{K}, p \in \mathcal{P} \right\},$$

(5.12)
5.2 An Accelerated Algorithm

\( H = \{ (\tilde{\beta}, y) | \tilde{\beta} \geq 1, y \geq 1 \} \),

(5.13)

with

\( \mathcal{P} = \{ p | 0 \leq p_i^k \leq P_{i}^{\text{max}}, \forall i \in \mathcal{L}, \forall k \in \mathcal{K} \} \).

(5.14)

With the above transformation, the joint scheduling and power control problem can be readily solved using the polyblock outer approximation algorithm. Interested readers are referred to [23], where the algorithm is named S-MAPEL, where the prefix “S” stands for scheduling.

5.2 An Accelerated Algorithm

There are \((n+1)^2\) variables in Problem (5.11), where \(n\) is the number of links in the system. In other words, the number of variables grows as \(O(n^2)\), and hence the computational complexity is high when \(n\) is large.

To see this, notice that in each iteration, the size of the vertex set \(\mathcal{T}_k\) of enclosing polyblocks grows in proportion to the number of variables in the problem. This will lead to a long convergence time if there are a large number of variables, as we need to compare all vertices in \(\mathcal{T}_k\) to choose the best one in each iteration.

To address the computational complexity problem, we present here an accelerated algorithm A-S-MAPEL (where the prefix “A” stands for accelerated) that expedites the algorithm by removing unnecessary vertices in each iteration. The intuition is as follows. Consider an optimal solution to Problem (5.11)

\( (\tilde{\beta}^*, y^*) = (\tilde{\beta}_1^*, \ldots, \tilde{\beta}_{n+1}^*, y_1^1, \ldots, y_n^1, \ldots, y_1^{n+1}, \ldots, y_n^{n+1}) \).

A closer look at the problem suggests that a new vector obtained by swapping the values of \((\tilde{\beta}_i^*, y_i^1, \ldots, y_i^n)\) and \((\tilde{\beta}_j^*, y_j^1, \ldots, y_j^n)\) for any pair of \(i\) and \(j\) is also an optimal solution. This is because the ordering of the time segments does not affect the sum data rate of each user, and hence does not affect the value of the utility functions.

This inherent symmetry suggests that we may delete some “symmetric” vertices of the enclosing polyblock without affecting the optimality of the polyblock outer approximation algorithm.
Suppose that
\[(\delta, z) = (\delta^1, \ldots, \delta^{n+1}, z_1^1, \ldots, z_{n+1}^1, \ldots, z_n^1, \ldots, z_{n+1}^n)\]
is an optimal vertex of the enclosing polyblock of a certain iteration. Consider a symmetric vertex of \((\delta, z)\), denoted by \((\hat{\delta}, \hat{z})\), which is obtained from \((\delta, z)\) by swapping the values of \((\delta^i, z_1^i, \ldots, z_n^i)\) and \((\delta^j, z_1^j, \ldots, z_n^j)\) for any pairs of \((i, j)\). Obviously, \(\Phi(\delta, z) = \Phi(\hat{\delta}, \hat{z})\). Simple calculation can show that the projection points of \((\delta, z)\) and \((\hat{\delta}, \hat{z})\) are still symmetric. Eventually, these symmetric vertices would lead to the same optimal objective function value just with different orderings for the time segments. Thus, for each optimal vertex \((\delta, z)\), we can remove all the symmetric vertices without affecting the optimality of the algorithm.

One difficulty in carrying out this idea lies in the identification of symmetric vertices from all vertices that yield the same optimal value at an iteration. To address this issue, A-S-MAPEL makes a simplifying assumption that all equally optimal vertices are symmetric vertices. Denote the equally optimal vertices at the \(n\)th iteration as \(Z_k = \{ (\delta, z) | \Phi((\delta, z)) = \Phi((\hat{\delta}, \hat{z})) \text{ and } (\delta, z) \in T_k \}\), where \((\hat{\delta}, \hat{z}) = \arg\max \{ \Phi((\delta, z)) | (\delta, z) \in T_k \}\). A-S-MAPEL then selects the one with the smallest difference between \(\Phi((\delta, z))\) and \(\Phi(\pi^G((\delta, z)))\), and deletes all other vertices in \(Z_k\).

For the practical implementation, we allow the existence of an error tolerance \(tol\) to further expedite the computational speed. Thus, the set \(Z_n\) is extended to
\[Z_k = \{ (\delta, z) | (1 + tol) \times \Phi((\delta, z)) \geq \Phi((\hat{\delta}, \hat{z})) \text{ and } (\delta, z) \in T_k \}\.
\]

### 5.3 Numerical Results

#### 5.3.1 Near Optimality of A-S-MAPEL

Now we investigate the performance of the A-S-MAPEL algorithm under different choices of algorithm parameters. We first consider a four-link network as shown in Figure 5.3, where \(d = 5\) m. Assume that the channel gain between the transmitter node \(i\) and the receiver node \(j\)
5.3 Numerical Results

Fig. 5.3 A network topology with four links.

is $d_{ij}^{-a}$, where $d_{ij}$ denotes the distance between the two nodes. Assume that $P_{\text{max}} = (1.0 \ 1.0 \ 1.0 \ 1.0)$ mW, and $n_i = 0.1 \mu W$ for all links. There are no minimum data rate constraints in this example. In Figure 5.4 and Figure 5.6, we investigate the system utility obtained by A-S-MAPEL under different error tolerances $\epsilon$ and $tol$, with different utility functions $U_i(r_i) = \log r_i$ and $U_i(r_i) = \frac{1}{1 + \exp(-r_i)}$, respectively. As a benchmark, the optimal system utilities are also plotted. Correspondingly, the numbers of iterations for convergence are plotted in Figures 5.5 and 5.7, respectively.

From Figures 5.4 and 5.6, we can see that the A-S-MAPEL algorithm achieves a result very close to the global optimal solution. For example, when $tol = 0.0005$, A-S-MAPEL obtains a system utility that is only 0.33% away from the optimum for proportional fair utility, and 0.17% away from the optimum for sigmoidal utility. On the other hand, it is not surprising to see that the algorithm performance improves with a smaller value of either $\epsilon$ or $tol$. In general, the algorithm performance is not sensitive to the value of $\epsilon$ when $tol$ is small enough. For example, when $tol = 0.0005$, the obtained system utility is roughly constant for any $\epsilon \in [0,0.5]$ for both proportional fair utility and sigmoidal utility. However, Figures 5.5 and 5.7 show that the convergence time of A-S-MAPEL can be quite different for different objective functions. Moreover, the total number of iterations increases when either $\epsilon$ or $tol$ decreases, and the increase is drastic when $tol$ is close to 0. Obviously, parameters $\epsilon$ and $tol$ provide a tuning knob for
5.3.2 Optimal power controlled scheduling versus Node Density

In this subsection, we vary $d$ in Figure 5.3 to investigate the effect of node density on the optimal power control scheduling. As an example,
5.3 Numerical Results

![Graph showing the summation of sigmoidal functions for different error tolerances.]

*Fig. 5.6* Summation of sigmoidal functions for different error tolerance $\epsilon$.

![Graph showing the number of iterations needed for different error tolerances.]

*Fig. 5.7* The number of iterations needed for different error tolerance $\epsilon$ when obtaining summation of sigmoidal functions.

we set $U_i(r_i) = \log r_i$, and let the minimum data rate constraints be 1.0 bps/Hz for all links. Other settings are the same as the previous example. In Figure 5.8, we let $d = 5, 10, 15$ m, and set the scheduling period to be 10 seconds for each $d$.

Figure 5.8 shows that the optimal power and scheduling solution heavily depend on the node density. Specifically, when the four links are close to each other (e.g., $d = 5$ m, from 0 to 10 seconds), the
optimal transmission power varies with time, implying that scheduling is an indispensable component in dense networks. On the other hand, scheduling is no longer necessary when the node density is small. For example, when \( d = 10 \text{ m} \) (i.e., from 10 to 20 seconds), the optimal transmission power of each link does not vary with time any more. Furthermore, when links are significantly far away from each other (e.g., \( d = 15 \text{ m} \), from 20 to 30 seconds), it is optimal to have all links transmit at the maximum power simultaneously.

5.3.3 Performance Study of On–off Power Control, Pure Power Control and On–off Scheduling

One key application of S-MAPEL and A-S-MAPEL is to provide a benchmark to evaluate the performance of other schemes. As an illustration, we evaluate the performance of three widely accepted schemes in the literature, namely pure power control, on–off power control, and on–off power control with scheduling (also referred to as on–off scheduling). In particular, the MAPEL algorithm discussed in Section 4 is used...
5.3 Numerical Results

to obtain the optimal power control solution. With the on–off power control, each transmitter either transmits at the maximum power level $P_{\text{max}}^i$ or remains silent. Meanwhile, the on–off scheduling is the same as power controlled scheduling except that transmitters either transmit at the maximum power $P_{\text{max}}^i$ or remains silent. It can be seen that Theorem 5.2 also applies to this case, and hence no more than $n + 1$ slots are needed to achieve the optimal performance of on–off scheduling.

We consider $n$-link networks. Links are randomly placed in a 15 m-by-15 m area. The length of each link is uniformly distributed within [1 m, 2 m]. Meanwhile, set $P_{\text{max}}^i = 1$ mW and $n_i = 0.1$ $\mu$W. In Table 5.1, the performance of optimal joint power control and scheduling, pure power control, on–off power control, and on–off scheduling are given for different utility functions when $n = 3$ and $n = 4$. Each value in the table is an average over 50 different topologies.

We can see that both power control schemes without scheduling are outperformed by the ones with scheduling. This is because in a dense network, power control alone is not sufficient to eliminate strong levels of interference between close-by links. One interesting observation is that without scheduling, on–off power control may lead to a

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Average performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$U_i(r_i) = \log(r_i)$</td>
</tr>
<tr>
<td></td>
<td>Three-link</td>
</tr>
<tr>
<td>On–off power control without</td>
<td>4.4476</td>
</tr>
<tr>
<td>scheduling</td>
<td></td>
</tr>
<tr>
<td>Pure power control</td>
<td>4.6801</td>
</tr>
<tr>
<td>On–off scheduling</td>
<td>5.1668</td>
</tr>
<tr>
<td>Power controlled scheduling</td>
<td>5.2276</td>
</tr>
</tbody>
</table>

Table 5.1. Comparison of pure power control, on–off scheduling versus power controlled scheduling.
much lower system utility compared with the optimal power control solution. In contrast, the performance gap between on–off scheduling and optimal power controlled scheduling is negligible. This is due to the fact that the links that are scheduled to transmit in the same time slot are typically far from each other and do not impose excessive interference on one another. As a result, it is likely to be optimal or very close to optimal for the links to transmit at the maximum power level. In practice, most off-the-shelf wireless devices are only allowed to either transmit at the maximum power (i.e., be on) or remain silent (i.e., be off). Therefore, scheduling is an indispensable component for system utility maximization if “off-the-shelf” wireless devices are to be used.
Optimal Transmit Beamforming in MISO Interference Channels

In this section, we generalize the discussion in Section 4 to multi-input-single-output (MISO) interference channels, where we jointly optimize the transmit power and transmit beamforming. The topic has been recently investigated in [12, 39]. We will show that the optimal transmit beamforming problem can be formulated as a nonconvex general quadratic fractional programming (GQFP), which can further be transformed to a monotonic optimization problem. Moreover, the upper boundary projection point can be obtained via bisection search and second order cone programming (SOCP).

6.1 System Model and Problem Formulation

Consider a wireless system with a set $\mathcal{L} = \{1, \ldots, n\}$ of distinct MISO links. Each link includes a multi-antenna transmitter $T_i$ and a single-antenna receiver $R_i$. The channel gain between node $T_i$ and node $R_j$ is denoted by vector $\mathbf{h}_{ij}$, which is determined by various factors such as path loss, shadowing, and fading effects. Let $\mathbf{w}_i$ denote the beamforming vector of $T_i$, which yields a transmit power level of $p_i = ||\mathbf{w}_i||^2_2$. Likewise, $\eta_i$ denotes the noise power at node $R_i$. Thus, the SINR of
link $i$ is

$$\gamma_i(w_1, \ldots, w_n) = \frac{|h_i^H w_i|^2}{\sum_{j \neq i} |h_j^H w_j|^2 + \eta_i}. \quad (6.1)$$

Like before, we use $U(\cdot)$ to denote the system utility function, which monotonically increases with $\gamma = (\gamma_i, \forall i \in \mathcal{L})$. We aim to solve the following utility maximization problem:

$$\max_{w_1, \ldots, w_n} U(\gamma) \quad (6.2a)$$

subject to

$$\gamma_i(w_1, \ldots, w_n) \geq \gamma_i, \min, \forall i = 1, \ldots, n, \quad (6.2b)$$

$$0 \leq ||w_i||^2 \leq P_i^{\text{max}}, \forall i = 1, \ldots, n, \quad (6.2c)$$

where $\gamma_i, \min$ is the minimum SINR requirement of link $i$ and $P_i^{\text{max}}$ is the maximum transmit power of link $i$.

Compared to the Formulation (4.2) in Section 4, we can see that Formulation (6.2) is as the GLFP defined in Definition 2.9, except that the nominator and denominator of $\gamma_i$ are quadratic functions. Indeed, Formulation (6.2) is a GQFP problem with quadratic constraints. Using the same approach as before, we can transform Problem (6.2) into a canonical monotonic optimization problem:

$$\max_y U(y) \quad (6.3)$$

subject to

$$y \in \mathcal{G} \cap \mathcal{H},$$

where $\mathcal{G} = \{ y | 0 \leq y_i \leq \gamma_i(w_1, \ldots, w_n), 0 \leq ||w_i||^2 \leq P_i^{\text{max}} \forall i \in \mathcal{L} \}$ and $\mathcal{H} = \{ y | y_i \geq \gamma_{i, \min} \forall i \in \mathcal{L} \}$.

### 6.2 Algorithm

As we have discussed before, a key step of the polyblock outer approximation algorithm is to find the projection point $\pi_{\mathcal{G}}(z_k)$ on the upper boundary of the feasible set. In Section 4, $\pi_{\mathcal{G}}(z_k)$ can be efficiently obtained by the bisection search algorithm, as checking the feasibility of an SINR vector $\bar{\alpha}z_k$ for SISO interference channels is as simple as solving an eigenvalue decomposition problem. In MISO channels, however, checking the feasibility of an SINR vector is not as simple.
To check whether a vector $\alpha z_k \in \mathcal{G}$, we solve the following problem

$$
\max_{w_1, \ldots, w_n} 0 \quad (6.4a)
$$

subject to

$$
\frac{|h_{ii}^H w_i|^2}{\sum_{j \neq i} |h_{ji}^H w_j|^2 + \eta_i} \geq \alpha z_{ki}, \forall i = 1, \ldots, n, \quad (6.4b)
$$

$$
0 \leq \|w_i\|^2 \leq P_{\text{max}}^i, \forall i = 1, \ldots, n. \quad (6.4c)
$$

The problem returns a value 0 if the constraints are feasible (i.e., $\alpha z_k$ is achievable) and $-\infty$ otherwise. At the first glance, Problem (6.4) is nonconvex due to the nonconvexity of the SINR constraints. Nonetheless, a close observation reveals that one can adjust the phase of $w_i$ to make $h_{ii}^H w_i$ real and nonnegative without affecting the value of $|h_{ii}^H w_i|$. Thus, without loss of generality, we can assume that $h_{ii}^H w_i$ is real and nonnegative for all $i$, Problem (6.4) becomes

$$
\max_{w_1, \ldots, w_n} 0 \quad (6.5a)
$$

subject to

$$
\sqrt{\alpha z_{ki}} \left\| \begin{array}{c}
h_{ii}^H w_1 \\
\vdots \\
h_{ni}^H w_n \\
\sqrt{\eta_i}
\end{array} \right\|_2 \leq h_{ii}^H w_i, \forall i = 1, \ldots, n \quad (6.5b)
$$

$$
0 \leq \|w_i\|^2 \leq P_{\text{max}}^i, \forall i = 1, \ldots, n. \quad (6.5c)
$$

which is an SOCP. With this, $\pi_G(z_k)$ can be found using the bisection method in Algorithm 1 where each iteration solves Problem (6.5) with a given $\alpha$.

Now that $\pi_G(z_k)$ can be obtained efficiently, we can solve the monotonic optimization problem in Equation (6.3) following Algorithm 2 (with possible enhancements discussed in Section 3.3). Once an optimal $y^*$ is obtained, the optimal $w_i^*$ can be recovered by solving a series of equations

$$
\frac{|h_{ii}^H w_i|^2}{\sum_{j \neq i} |h_{ji}^H w_j|^2 + \eta_i} = y_i^*, \forall i = 1, \ldots, n. \quad (6.6)
$$

One way to solve the equations is to again solve the SOCP Problem (6.5) with $\alpha z_{ki}$ replaced by $y_i^*$. Since $y^*$ is on the upper
boundary of the feasible set, the $w_i$'s obtained would satisfy Problem (6.5b) with equality, and thus is also a solution to Problem (6.6).

6.3 Extensions

6.3.1 MISO Multicasting

Problem (6.4) can be transformed to an SOCP problem, only because the numerator of the quadratic fractional function (Problem (6.4b)) can be converted to a real number by rotating the phase of $w_i$. This is not always possible for GQFP problems in general. As an example, consider a slightly different system, where each multi-antenna transmitter multicasts to a number of single-antenna receivers. Denote the set of transmitters as $\mathcal{T} = \{1, \ldots, n\}$, and the multicast group of transmitter $i$ as $\mathcal{M}(i)$. Then, the SINR received by a receiver $k \in \mathcal{M}(i)$ is

$$\gamma_k(w_1, \ldots, w_n) = \frac{|h_{ik}^H w_i|^2}{\sum_{j \in \mathcal{T}, j \neq i} |h_{jk}^H w_j|^2 + \eta_k},$$

and the utility maximization problem becomes

$$\max_{w_1, \ldots, w_n} U(\gamma) \quad \text{(6.7a)}$$

subject to

$$\gamma_k(w_1, \ldots, w_n) \geq \gamma_{k, \min}, \forall k \in \mathcal{M}(1) \cup \cdots \cup \mathcal{M}(n),$$

$$0 \leq |w_i|^2 \leq P_i^{\max}, \forall i = 1, \ldots, n. \quad \text{(6.7b)}$$

Similar to the procedures in Section 6.2, the following problem needs to be solved to check whether a vector $\tilde{\alpha} z \in \mathcal{G}$.

$$\max_{w_1, \ldots, w_n} 0 \quad \text{(6.8a)}$$

subject to

$$\frac{|h_{ik}^H w_i|^2}{\sum_{j \in \mathcal{T}, j \neq i} |h_{jk}^H w_j|^2 + \eta_k} \geq \tilde{\alpha} z_k, \forall k \in \mathcal{M}(1) \cup \cdots \cup \mathcal{M}(n),$$

$$0 \leq |w_i|^2 \leq P_i^{\max}, \forall i = 1, \ldots, n. \quad \text{(6.8b)}$$

We notice that it is not possible to adjust the phase of $w_i$ so that $h_{ik}^H w_i$ is real for all $k$. Thus, Problem (6.8) is a nonconvex quadratic
programming (QP) and cannot be converted to an SOCP. One way to tackle the problem is through the semidefinite programming (SDP) relaxation that replaces $|h_{ik}^H w_i|^2$ by $\text{tr}(H_{ik} W_i)$, where $H_{ik} = h_{ik} h_{ik}^H$ and $W_i = w_i w_i^H$, and ignore the constraint that $W_i$ is a rank-one matrix. This, however, may lead to infeasible solutions when the resultant $W_i$’s cannot be reduced to rank-one matrices. Thus, it remains an interesting and challenging problem to solve the monotonic optimization Problem (6.7) efficiently and optimally.

6.3.2 SIMO Interference Channel

Let us slightly revise the system configuration such that each transmitter has a single antenna and each receiver has multiple antennas. Let $q_i$ and $p_i$ be the receive beamforming vector and the transmit power for the $i^{th}$ link, respectively. The SINR of link $i$ is

$$\gamma_i(q_1, \ldots, q_n, p_1, \ldots, p_n) = \frac{p_i |q_i^H h_{ii}|^2}{q_i^H \left( \sum_{j \neq i} p_j h_{ji} h_{ji}^H + \eta_i I \right) q_i}.$$  \hspace{1cm} (6.9)

The feasibility an SINR vector $\bar{\alpha} z_k$ can be checked by the following problem:

$$\max_{q_1, \ldots, q_n, p_1, \ldots, p_n} 0 \hspace{1cm} (6.10a)$$

subject to

$$\frac{p_i |q_i^H h_{ii}|^2}{q_i^H \left( \sum_{j \neq i} p_j h_{ji} h_{ji}^H + \eta_i I \right) q_i} \geq \bar{\alpha} z_i, \forall i, \hspace{1cm} (6.10b)$$

$$0 \leq p_i \leq P_{i}^{\max}, \forall i. \hspace{1cm} (6.10c)$$

The problem is equivalent to the max–min problem

$$\max_{q_1, \ldots, q_n \in P} \min_i \frac{p_i |q_i^H h_{ii}|^2}{\bar{\alpha} z_i q_i^H \left( \sum_{j \neq i} p_j h_{ji} h_{ji}^H + \eta_i I \right) q_i} \hspace{1cm} (6.11a)$$

subject to

$$0 \leq p_i \leq P_{i}^{\max}, \forall i. \hspace{1cm} (6.11b)$$

which indicates the feasibility of $\bar{\alpha} z_k$ if the optimal value is larger than 1. Due to the product form of $q_i$’s and $p_i$’s, Problems (6.10) and (6.11) are more intractable than the one encountered in MISO interference channel.
A recent attempt to solve Problem (6.11) was made in [16], which proposed an iterative algorithm that optimizes $q_i$’s and $p$ alternatively. Instead of approaching Problem (6.11) directly, [16] solves $n$ subproblems. In particular, the $l$th subproblem is obtained by replacing the $n$ constraints in Problem (6.11b) with an equality power constraint of user $l$, i.e.,

$$p_l = P_l^{\text{max}}.$$ 

The rationale behind is that the optimal solution to Problem (6.11) would satisfy at least one constraint in Problem (6.11b) with equality. Thus, the solution to a subproblem would be the same as that to Problem (6.11), if it turns out to satisfy all the $n$ constraints in Problem (6.11b).

To solve the $l$th subproblem, notice that for given $p$, the optimal $q_i$’s are minimum-mean-squared-error (MMSE) beamforming vectors, i.e.,

$$q_i = \left( \sum_{j \neq i} p_j h_{ji} h_{ji}^H + \eta_i I \right)^{-1} h_{ii}.$$  

(6.12)

Moreover, for given $q_i$’s, the optimal $p_{\text{ext}} := [p, 1]^T$ is obtained as the dominant eigenvector of matrix

$$A_i = \begin{bmatrix} B & u \\ \frac{1}{\|e_i\|^2} e_i^T B & \frac{1}{\|e_i\|^2} e_i^T u \end{bmatrix},$$  

(6.13)

where

$$B_{ij} = \begin{cases} 0, & i = j \\ \frac{\bar{\alpha}_{z_k} |q_i^H h_{ji}|^2}{|q_i^H h_{ii}|^2}, & i \neq j \end{cases}$$  

(6.14)

$$u_i = \frac{\bar{\alpha}_{z_k} \eta_i |q_i|^2}{|q_i^H h_{ii}|^2},$$  

(6.15)

and $e_i$ is the $l$th standard unit vector. Authors in [16] showed that the optimal solution to the problem can be obtained by optimizing $q_i$’s and $p$ alternatively. Readers are referred to [16] for details.
7

Optimal Random
Medium Access Control (MAC)

There are two major types of wireless medium access control (MAC) protocols: scheduling-based (e.g., in cellular systems) and contention-based (e.g., in wireless local area networks) [20]. The sample applications in the last three sections mainly arise from scheduling-based MAC protocols. Therein, the objective and constraint functions are increasing functions of fractional functions. In this section, we are going to focus on contention-based random access networks, where the objective and constraint functions are increasing functions of polynomial functions.

7.1 System Model and Problem Formulation

We consider a slotted ALOHA network with $n$ competing links. At each time slot, a node $i$ attempts to access the channel by transmitting a packet with probability $\theta_i$. The packet can be decoded correctly at the receiver if only one node transmits. Otherwise, a collision occurs and all packets are corrupted.

In such a network, the average throughput of node $i$ is calculated as

$$r_i = c_i \theta_i \prod_{j \neq i} (1 - \theta_j),$$

(7.1)

where $c_i$ is the data rate at which node $i$ transmits a packet.
We wish to find the optimal transmission probabilities \( \theta = (\theta_i, \forall i = 1, \ldots, n) \) such that the overall system utility is maximized. This is mathematically formulated as

\[
\begin{align*}
\max_{\theta} & \quad U(r(\theta)) \\
\text{subject to} & \quad r_i(\theta) \geq r_{i, \min} \forall i = 1, \ldots, n \\
& \quad 0 \leq \theta_i \leq 1, \forall i = 1, \ldots, n,
\end{align*}
\]

where \( r = (r_i, \forall i = 1, \ldots, n) \). \( U(\cdot) \) is an increasing function that is not necessarily concave. Specifically, we can choose \( U(r(\theta)) = \sum_i r_i(\theta) \) to maximize the total system throughput, \( U(r(\theta)) = \sum_i \log r_i(\theta) \) or equivalently \( U(r(\theta)) = \prod_i r_i(\theta) \) to maximize proportional fairness, and \( U(r(\theta)) = \min_i r_i(\theta) \) to maximize max–min fairness.

The objective and constraint functions of Problem (7.2) are increasing functions of polynomial functions of \( \theta \), which are not monotonic in \( \theta \). In the next section, we will first show that by introducing \( n \) auxiliary variables, the problem can be transformed to a canonical monotonic optimization problem with \( 2n \) variables. Then, we will show that the number of variables can be further reduced to \( n \), by combining geometric programming with monotonic programming.

### 7.2 Algorithm

#### 7.2.1 Transformation to a Canonical Monotonic Optimization Problem

Define \( \hat{\theta}_i = 1 - \theta_i \). Substituting it into Equation (7.2), we have

\[
\begin{align*}
\max_{\theta, \hat{\theta}} & \quad U((\theta, \hat{\theta})) = U\left(c_1 \theta_1 \prod_{j \neq 1} \hat{\theta}_j, \ldots, c_n \theta_n \prod_{j \neq n} \hat{\theta}_j\right) \\
\text{subject to} & \quad c_i \theta_i \prod_{j \neq i} \hat{\theta}_j \geq r_{i, \min} \forall i = 1, \ldots, n \\
& \quad \theta_i + \hat{\theta}_i \leq 1, \forall i = 1, \ldots, n, \\
& \quad \theta_i \geq 0, \forall i = 1, \ldots, n, \\
& \quad \hat{\theta}_i \geq 0, \forall i = 1, \ldots, n.
\end{align*}
\]
Now, the objective and constraint functions are all monotonically increasing functions of \( \theta_i \)'s and \( \hat{\theta}_i \)'s. Notice that in Equation (7.3c), we have replaced the constraint \( \theta_i + \hat{\theta}_i = 1 \) by \( \theta_i + \hat{\theta}_i \leq 1 \). This, however, is not a relaxation, as the optimal solution will always occur at the upper boundary of the feasible region, i.e., when the inequality in Equation (7.3c) is satisfied with equality.

One can easily recognize Equation (7.3) as a monotonic optimization problem. Indeed, it can be readily written into the following canonical form:

\[
\max \{ U((\theta, \hat{\theta})) | (\theta, \hat{\theta}) \in \mathcal{G} \cap \mathcal{H} \}, \tag{7.4}
\]

where

\[
\mathcal{G} = \{ (\theta, \hat{\theta}) | \theta_i + \hat{\theta}_i \leq 1, \forall i = 1, \ldots, n \},
\]

\[
\mathcal{H} = \{ (\theta, \hat{\theta}) | c_i \theta_i \prod_{j \neq i} \hat{\theta}_j \geq r_i, \min, \theta_i \geq 0, \hat{\theta}_i \geq 0, \forall i = 1, \ldots, n \}.
\]

It turns out that the polyblock outer approximation algorithm can be implemented efficiently to solve this problem. First of all, the initial enclosing polyblock can be simply constructed as \([0, 2^n], [1, 2^n] \), where \(0_{2^n}\) and \(1_{2^n}\) are vectors of zeros and ones with size \(2^n\), respectively. Secondly, the projection point on the upper boundary of \( \mathcal{G} \) can be simply calculated without the need of a bisection algorithm. Suppose that \((z_k, \hat{z}_k)\) is the polyblock vertex that maximizes the objective function value. The projection point is given by \( \alpha(z_k, \hat{z}_k) \), where \( \alpha \) can be straightforwardly obtained by

\[
\alpha = \max \{ \alpha | (z_k, \hat{z}_k) \in \mathcal{G} \} \tag{7.5}
\]

\[
\alpha = \max \{ \alpha | (z_k + \hat{z}_k) \leq 1 \ \forall i \}\]

\[
\alpha = \min_i \frac{1}{z_k + \hat{z}_k}.
\]

### 7.2.2 Problem Size Reduction

The transformation in Subsection 7.2.1 increases the number of variables in Problem (7.4) from \(n\) to \(2n\), due to the auxiliary variables \( \hat{\theta} \). As we discussed before, the complexity of monotonic optimization may
increase drastically as the problem size becomes large. Thus, one should cautiously reduce the size of the problem as much as possible. In view of this, we propose to transform Problem (7.4) into the following monotonic optimization problem, where the number of variables is reduced back to $n$.

$$
\max_y U(y) = U(c_1y_1, \ldots, c_ny_n) \tag{7.6a}
$$

subject to \quad y \in G \cap H, \tag{7.6b}

where

$$
G = \{ y | 0 \leq y_i \leq \theta_i \prod_{j \neq i} \tilde{\theta}_j, \theta_i + \hat{\theta}_i \leq 1, \theta_i \geq 0, \hat{\theta}_i \geq 0, \forall i \},
$$

$$
H = \{ y | c_iy_i \geq r_i,_{\min}, \forall i \}.
$$

Like before, the problem can be efficiently solved using the polyblock outer approximation algorithm in Algorithm 2, as long as the upper boundary projection point can be calculated with reasonable computational complexity. Indeed, the following discussion indicates that the projection point can be obtained by solving a convex optimization problem with reasonable computational complexity.

Suppose that $z_k$ is the polyblock vertex that maximizes the objective function value in the $k^{th}$ iteration, then the projection point $\alpha z_k$ can be obtained by solving the following problem.

$$
\max_{\alpha, \theta} \quad \alpha \tag{7.7a}
$$

subject to \quad $\alpha z_k \leq c_i\theta_i \prod_{j \neq i} \tilde{\theta}_j \quad \forall i \tag{7.7b}$

$$
\theta_i + \hat{\theta}_i \leq 1 \quad \forall i \tag{7.7c}
$$

$$
\theta_i \geq 0, \hat{\theta}_i \geq 0 \quad \forall i. \tag{7.7d}
$$

This problem can be equivalently rewritten as a standard Geometric Programming problem:

$$
\min_{\alpha, \theta} \quad \alpha^{-1} \tag{7.8a}
$$

subject to \quad $c_i^{-1}\alpha z_k \theta_i^{-1} \prod_{j \neq i} \tilde{\theta}_j^{-1} \leq 1 \quad \forall i \tag{7.8b}$
\[ \theta_i + \hat{\theta}_i \leq 1 \forall i \]  
\[ \theta_i \geq 0, \hat{\theta}_i \geq 0 \forall i, \]  
which can be further transformed to a convex optimization problem \[2\].

### 7.3 Discussions

So far, we have assumed that the \( n \) links fully interfere with each other, i.e., no two links can be simultaneously active. The problem can be easily extended to cases when a link only interferes with, and is interfered by, nearby links. In this case, the throughput calculation in Equation (7.1) is modified as

\[ r_i = c_i \theta_i \prod_{j \in I(i)} (1 - \theta_j), \]  

where \( I(i) \) denotes the set of links that interfere with link \( i \). All other formulations remain the same.

There have been numerous work on the throughput performance of random-access networks, e.g., carrier-sensing multi-access (CSMA) networks. While most of the early work focuses on fully interfered networks \[1, 26, 42\], the analysis on non-fully interfered networks is much more complicated (and interesting) \[6, 7, 14, 21\]. A common challenge here is to understand the ultimate throughput that is achievable by random access, and to understand how to tune the parameters of the CSMA protocol to achieve the maximum throughput. The monotonic optimization method in this section serves as a useful tool to address this challenge, as we can now easily calculate the maximum system utility, including throughput, of random-access networks.

Noticingly, the monotonic optimization method only quantifies the ultimate performance that is achievable, but does not indicate how to achieve such performance. This may spur interesting future research interests in reverse engineering the CSMA protocol to find the optimal parameters that maximize a system utility.
The main purpose of this monograph is to introduce the framework of monotonic optimization to the research community of communication and networking. Instead of giving rigorous mathematical proofs, we put more emphasis on illustrative applications, with the hope that the monograph is more accessible to a general audience. For interested readers, complete mathematical proofs can be found in [22, 31, 33, 36, 37].

Many global optimization problems in engineering systems exhibit monotonic or hidden monotonic structures. In Section 2, we have introduced techniques to formulate such problems into canonical monotonic optimization problems, which maximize (or minimize) an increasing function over normal sets. In Section 3, we have presented the polyblock outer approximation algorithm, which solves the monotonic optimization problem by gradually refining the approximation of the feasible set by a nested sequence of polyblocks. An important issue of such an algorithm is its computational complexity, which heavily relies on the complexity of computing the upper boundary projection $\pi_G(z)$ of a vertex $z$. In some special cases, such as the problem encountered in Section 7, the projection can be explicitly calculated or obtained through convex optimization. In general, one can obtain the projection point
through the bisection search, where a feasibility check is performed in each iteration. In this case, the algorithm is efficient only when the feasibility check can be carried out efficiently.

Through different applications, we have illustrated various techniques to expedite the monotonic optimization algorithm. To this end, we often need to incorporate the domain knowledge of the underlying system. For example, in Section 4 we have turned to Perron Frobenius theorem for checking the feasibility of a power control problem. In Section 5 we have made use of the symmetry of time intervals to accelerate the algorithm. In Section 6 the seemingly nonconvex feasibility check problem is converted to an SOCP, which greatly reduces the complexity to obtain the upper boundary projection. We hope that these examples will trigger new research interests to identify more useful problem structures in communication and networking systems.

The main benefit of using monotonic programming is to solve the global optimal solution in a centralized fashion. Very often, the global optimal solution is used as a performance benchmark for evaluating other low complexity heuristic algorithms, and thus the algorithm complexity is not a big issue. On the other hand, computational complexity becomes a primary concern, if we wish to use the method for real-time network control. In this case, we may either go for heuristic algorithms or exploit other problem structures besides monotonicity. Such structures may include, for example, the objective and constraint functions being polynomial functions \cite{28, 29}, quasiconvex or quasiconcave functions \cite{18}, difference of convex functions \cite{30}, indefinite quadratic functions \cite{8}, and piecewise linear functions \cite{40}. This is an underexplored research area, and we hope that this monograph can inspire more exciting research along this direction.
References


References


References


