Mathematical Programming and Operations Research

Modeling, Algorithms, and Complexity

Examples in Excel and Python

(Work in progress)

Edited by: Robert Hildebrand

Contributors: Robert Hildebrand, Laurent Poirrier, Douglas Bish, Diego Moran

Version Compilation date: September 27, 2023

Preface

This entire book is a working manuscript. The first draft of the book is yet to be completed.

This book is being written and compiled using a number of open source materials. We will strive to properly cite all resources used and give references on where to find these resources. Although the material used in this book comes from a variety of licences, everything used here will be CC-BY-SA 4.0 compatible, and hence, the entire book will fall under a CC-BY-SA 4.0 license.

MAJOR ACKNOWLEDGEMENTS

I would like to acknowledge that substantial parts of this book were borrowed under a CC-BY-SA license. These substantial pieces include:

- "A First Course in Linear Algebra" by Lyryx Learning (based on original text by Ken Kuttler). A majority of their formatting was used along with selected sections that make up the appendix sections on linear algebra. We are extremely grateful to Lyryx for sharing their files with us. They do an amazing job compiling their books and the templates and formatting that we have borrowed here clearly took a lot of work to set up. Thank you for sharing all of this material to make structuring and formating this book much easier! See subsequent page for list of contributors.
- "Foundations of Applied Mathmatics" with many contributors. See `https://github.com/Foundations-of-Applied-Mathematics`. Several sections from these notes were used along with some formatting. Some of this content has been edited or rearranged to suit the needs of this book. This content comes with some great references to code and nice formatting to present code within the book. See subsequent page with list of contributors.
- "Linear Inequalities and Linear Programming" by Kevin Cheung. See `https://github.com/dataopt/lineqlpbook`. These notes are posted on GitHub in a ".Rmd" format for nice reading online. This content was converted to LaTeXusing Pandoc. These notes make up a substantial section of the Linear Programming part of this book.
- Linear Programming notes by Douglas Bish. These notes also make up a substantial section of the Linear Programming part of this book.

I would also like to acknowledge Laurent Porrier and Diego Moran for contributing various notes on linear and integer programming.

I would also like to thank Jamie Fravel for helping to edit this book and for contributing chapters, examples, and code.

Contents

Introduction

Letter to instructors

This is an introductory book for students to learn optimization theory, tools, and applications. The two main goals are (1) students are able to apply the of optimization in their future work or research (2) students understand the concepts underlying optimization solver and use this knowledge to use solvers effectively.

This book was based on a sequence of course at Virginia Tech in the Industrial and Systems Engineering Department. The courses are *Deterministic Operations Research I* and *Deterministic Operations Reserach II*. The first course focuses on linear programming, while the second course covers integer programming an nonlinear programming. As such, the content in this book is meant to cover 2 or more full courses in optimization.

The book is designed to be read in a linear fashion. That said, many of the chapters are mostly independent and could be rearrranged, removed, or shortened as desited. This is an open source textbook, so use it as you like. You are encouraged to share adaptations and improvements so that this work can evolve over time.

Letter to students

This book is designed to be a resource for students interested in learning what optimizaiton is, how it works, and how you can apply it in your future career. The main focus is being able to apply the techniques of optimization to problems using computer technology while understanding (at least at a high level) what the computer is doing and what you can claim about the output from the computer.

Quite importantly, keep in mind that when someone claims to have *optimized* a problem, we want to know what kind of gaurantees they have about how good their solution is. Far too often, the solution that is provided is suboptimal by 10%, 20%, or even more. This can mean spending excess amounts of money, time, or energy that could have been saved. And when problems are at a large scale, this can easily result in millions of dollars in savings.

For this reason, we will learn the perspective of *mathematical programming* (a.k.a. mathematical optimization). The key to this study is that we provide gaurantees on how good a solution is to a given problem. We will also study how difficult a problem is to solve. This will help us know (a) how long it might take to solve it and (b) how good a of a solution we might expect to be able to find in reasonable amount of time.

We will later study *heuristic* methods - these methods typically do not come with gaurantees, but tend to help find quality solutions.

Note: Although there is some computer programming required in this work, this is not a course on programming. Thanks to the fantastic modelling packages available these days, we are able to solve complicated problems with little programming effort. The key skill we will need to learn is *mathematical modeling*: converting words and ideas into numbers and variables in order to communicate problems to a compter so that it can solve a problem for you.

As a main element of this book, we would like to make the process of using code and software as easy as possible. Attached to most examples in the book, there will be links to code that implements and solves the problem using several different tools from Excel and Python. These examples can should make it easy to solve a similar problem with different data, or more generally, can serve as a basis for solving related problems with similar structure.

How to use this book

Skim ahead. We recommend that before you come across a topic in lecture, that you skim the relevant sections ahead of time to get a broad overview of what is to come. This may take only a fraction of the time that it may take for you to read it.

Read the expected outcomes. At the beginning of each section, there will be a list of expected outcomes from that section. Review these outcomes before reading the section to help guide you through what is most relevant for you to take away from the material. This will also provide a brief look into what is to follow in that section.

Read the text. Read carefully the text to understand the problems and techniques. We will try to provide a plethora of examples, therefore, depending on your understanding of a topic, you many need to go carefully over all of the examples.

Explore the resources. Lastly, we recognize that there are many alternative methods of learning given the massive amounts of information and resources online. Thus, at the end of each section, there will be a number of superb resources that available on the internet in different formats. There are other free textbooks, informational websites, and also number of fantastic videos posted to youtube. We encourage to explore the resources to get another perspective on the material or to hear/read it taught from a differnt point of view or in presentation style.

Outline of this book

This book is divided in to 4 Parts:

Part I Linear Programming,

Part II Integer Programming,

Part III Discrete Algorithms,

There are also a number of chapters of background material in the Appendix.

The content of this book is desigened to encompass 2-3 full semester courses in an industrial engineering department.

Work in progress

This book is still a work in progress, so please feel free to send feedback, questions, comments, and edits to Robert Hildebrand at open.optimization@gmail.com.

1. Resources and Notation

Here are a list of resources that may be useful as alternative references or additional references.

Free notes and textbooks

- Mathematical Programming with Julia by Richard Lusby & Thomas Stidsen
- Linear Programming by K.J. Mtetwa, David
- A first course in optimization by Jon Lee
- Introduction to Optimizaiton Notes by Komei Fukuda
- Convex Optimization by Bord and Vandenberghe
- LP notes of Michel Goemans from MIT
- Understanding and Using Linear Programming - Matousek and Gärtner [Downloadable from Springer with University account]
- Operations Research Problems Statements and Solutions - Raúl PolerJosefa Mula Manuel Díaz-Madroñero [Downloadable from Springer with University account]

Notes, books, and videos by various solver groups

- AIMMS Optimization Modeling
- Optimization Modeling with LINGO by Linus Schrage
- The AMPL Book
- Microsoft Excel 2019 Data Analysis and Business Modeling, Sixth Edition, by Wayne Winston - Available to read for free as an e-book through Virginia Tech library at Orielly.com.
- Lesson files for the Winston Book
- Video instructions for solver and an example workbook
- youtube-OR-course

Gurobi Links

- Go to `https://github.com/Gurobi` and download the example files.
- Essential ingredients
- Gurobi Linear Programming tutorial
- Gurobi tutorial MILP
- GUROBI - Python 1 - Modeling with GUROBI in Python
- GUROBI - Python II: Advanced Algebraic Modeling with Python and Gurobi
- GUROBI - Python III: Optimization and Heuristics
- Webinar Materials
- GUROBI Tutorials

How to prove things

- Hammack - Book of Proof

Statistics

- Open Stax - Introductory Statistics

Linear Algebra

- Beezer - A first course in linear algebra
- Selinger - Linear Algebra
- Cherney, Denton, Thomas, Waldron - Linear Algebra

Real Analysis

- Mathematical Analysis I by Elias Zakon

Discrete Mathematics, Graphs, Algorithms, and Combinatorics

- Levin - Discrete Mathematics - An Open Introduction, 3rd edition
- Github - Discrete Mathematics: an Open Introduction CC BY SA
- Keller, Trotter - Applied Combinatorics (CC-BY-SA 4.0)
- Keller - Github - Applied Combinatorics

Programming with Python

- A Byte of Python
- Github - Byte of Python (CC-BY-SA)

Also, go to `https://github.com/open-optimization/open-optimization-or-examples` to look at more examples.

Notation

- 1 - a vector of all ones (the size of the vector depends on context)
- $\forall$ - for all
- $\exists$ - there exists
- $\in$ - in
- $\therefore$ - therefore
- $\Rightarrow$ - implies
- s.t. - such that (or sometimes "subject to".... from context?)
- $\{0,1\}$ - the set of numbers 0 and 1
- $\mathbb{Z}$ - the set of integers (e.g. $1,2,3,-1,-2,-3,...$)
- $\mathbb{Q}$ - the set of rational numbers (numbers that can be written as p/q for $p,q \in \mathbb{Z}$ (e.g. $1,1/6,27/2$)
- $\mathbb{R}$ - the set of all real numbers (e.g. 1, 1.5, π, e, $-11/5$)
- $\setminus$ - setminus, (e.g. $\{0,1,2,3\} \setminus \{0,3\} = \{1,2\}$)
- $\cup$ - union (e.g. $\{1,2\} \cup \{3,5\} = \{1,2,3,5\}$
- $\cap$ - intersection (e.g. $\{1,2,3,4\} \cap \{3,4,5,6\} = \{3,4\}$)
- $\{0,1\}^4$ - the set of 4 dimensional vectors taking values 0 or 1, (e.g. $[0,0,1,0]$ or $[1,1,1,1]$)
- $\mathbb{Z}^4$ - the set of 4 dimensional vectors taking integer values (e.g., $[1,-5,17,3]$ or $[6,2,-3,-11]$)

- $\mathbb{Q}^4$ - the set of 4 dimensional vectors taking rational values (e.g. $[1.5, 3.4, -2.4, 2]$)
- $\mathbb{R}^4$ - the set of 4 dimensional vectors taking real values (e.g. $[3, \pi, -e, \sqrt{2}]$)
- $\sum_{i=1}^{4} i = 1+2+3+4$
- $\sum_{i=1}^{4} i^2 = 1^2+2^2+3^2+4^4$
- $\sum_{i=1}^{4} x_i = x_1+x_2+x_3+x_4$
- $\square$ - this is a typical Q.E.D. symbol that you put at the end of a proof meaning "I proved it."
- For $x, y \in \mathbb{R}^3$, the following are equivalent (note, in other contexts, these notations can mean different things)
  - $x^\top y$ *matrix multiplication*
  - $x \cdot y$ *dot product*
  - $\langle x, y \rangle$ *inner product*

  and evaluate to $\sum_{i=1}^{3} x_i y_i = x_1 y_1 + x_2 y_2 + x_3 y_3$.

A sample sentence:

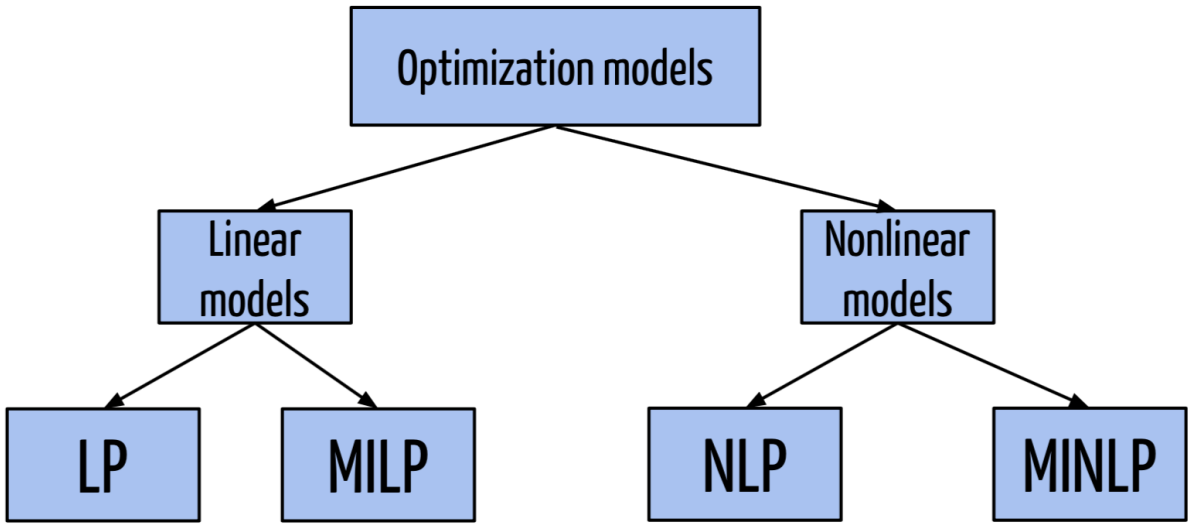
$$\forall x \in \mathbb{Q}^n \; \exists y \in \mathbb{Z}^n \setminus \{0\}^n s.t. x^\top y \in \{0, 1\}$$

"For all non-zero rational vectors x in n-dimensions, there exists a non-zero n-dimensional integer vector y such that the dot product of x with y evaluates to either 0 or 1."

2. Mathematical Programming

Outcomes

We will state main general problem classes to be associated with in these notes. These are Linear Programming (LP), Mixed-Integer Linear Programming (MILP), Non-Linear Programming (NLP), and Mixed-Integer Non-Linear Programming (MINLP).



© **problem-class-diagram**[1]

Figure 2.1: problem-class-diagram

Along with each problem class, we will associate a complexity class for the general version of the problem. See **??** for a discussion of complexity classes. Although we will often state that input data for a problem comes from $\mathbb{R}$, when we discuss complexity of such a problem, we actually mean that the data is rational, i.e., from $\mathbb{Q}$, and is given in binary encoding.

[1] **problem-class-diagram**, from **problem-class-diagram**. **problem-class-diagram**, **problem-class-diagram**.

2.1 Linear Programming (LP)

Some linear programming background, theory, and examples will be provided in **??**.

Linear Programming (LP):

Polynomial time (P)

Given a matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^n$, the *linear programming* problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2.1}$$

Linear programming can come in several forms, whether we are maximizing or minimizing, or if the constraints are $\leq, =$ or $\geq$. One form commonly used is *Standard Form* given as

Linear Programming (LP) Standard Form:

Polynomial time (P)

Given a matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^n$, the *linear programming* problem in *standard form* is

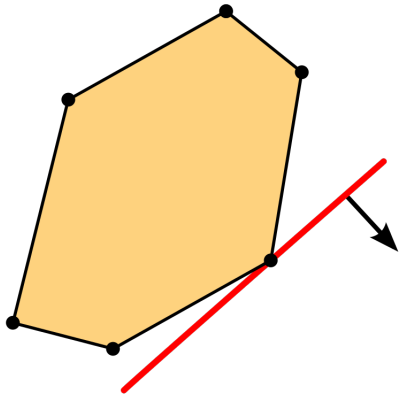
$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.2}$$

Figure 2.2

Exercise 2.1:

Start with a problem in form given as (2.1) and convert it to standard form (2.2) by adding at most m many new variables and by enlarging the constraint matrix A by at most m new columns.

[2] **wiki/File/linear-programming.png**, from **wiki/File/linear-programming.png**. **wiki/File/linear-programming.png**, **wiki/File/linear-programming.png**.



© wiki/File/linear-programming.png[2]
Figure 2.2: Linear programming constraints and objective.

2.2 Mixed-Integer Linear Programming (MILP)

Mixed-integer linear programming will be the focus of Sections **??**,**??**, **??**, and **??**. Recall that the notation $\mathbb{Z}$ means the set of integers and the set $\mathbb{R}$ means the set of real numbers. The first problem of interest here is a *binary integer program* (BIP) where all n variables are binary (either 0 or 1).

Binary Integer programming (BIP):

NP-Complete

Given a matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^n$, the *binary integer programming* problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \{0,1\}^n \end{aligned} \tag{2.1}$$

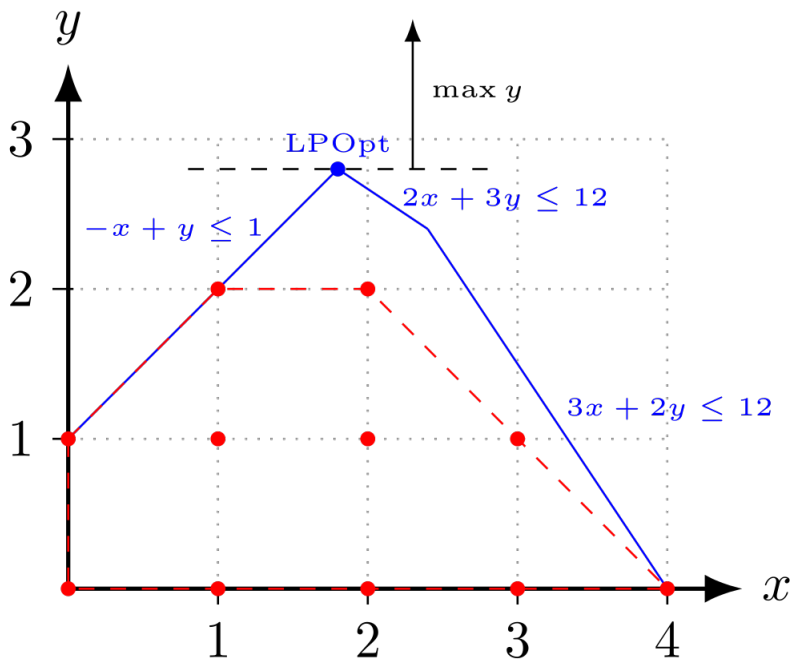
A slightly more general class is the class of *Integer Linear Programs* (ILP). Often this is referred to as *Integer Program* (IP), although this term could leave open the possibility of non-linear parts.

Figure 2.3

Integer Linear Programming (ILP):

NP-Complete

[3] **wiki/File/integer-programming.png**, from **wiki/File/integer-programming.png**. **wiki/File/integer-programming.png**, **wiki/File/integer-programming.png**.



© wiki/File/integer-programming.png[3]

Figure 2.3: Comparing the LP relaxation to the IP solutions.

Given a matrix $A \in \mathbb{R}^{m\times n}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^n$, the *integer linear programming* problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned} \tag{2.2}$$

An even more general class is *Mixed-Integer Linear Programming (MILP)*. This is where we have n integer variables $x_1,\ldots,x_n \in \mathbb{Z}$ and d continuous variables $x_{n+1},\ldots,x_{n+d} \in \mathbb{R}$. Succinctly, we can write this as $x \in \mathbb{Z}^n \times \mathbb{R}^d$, where $\times$ stands for the *cross-product* between two spaces.

Below, the matrix A now has $n+d$ columns, that is, $A \in \mathbb{R}^{m\times n+d}$. Also note that we have not explicitly enforced non-negativity on the variables. If there are non-negativity restrictions, this can be assumed to be a part of the inequality description $Ax \leq b$.

Mixed-Integer Linear Programming (MILP):

NP-Complete

Given a matrix $A \in \mathbb{R}^{m\times(n+d)}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^{n+d}$, the *mixed-integer linear program-*

ming problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n \times \mathbb{R}^d \end{aligned} \tag{2.3}$$

2.3 Non-Linear Programming (NLP)

NLP:

NP-Hard

Given a function $f(x)\colon \mathbb{R}^d \to \mathbb{R}$ and other functions $f_i(x)\colon \mathbb{R}^d \to \mathbb{R}$ for $i = 1,\ldots,m$, the *nonlinear programming* problem is

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad \text{for } i = 1,\ldots,m \\ & x \in \mathbb{R}^d \end{aligned} \tag{2.1}$$

Nonlinear programming can be separated into convex programming and non-convex programming. These two are very different beasts and it is important to distinguish between the two.

2.3.1. Convex Programming

Here the functions are all **convex!**

Convex Programming:

Polynomial time (P) (typically)

Given a convex function $f(x)\colon \mathbb{R}^d \to \mathbb{R}$ and convex functions $f_i(x)\colon \mathbb{R}^d \to \mathbb{R}$ for $i = 1,\ldots,m$, the *convex programming* problem is

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad \text{for } i = 1,\ldots,m \\ & x \in \mathbb{R}^d \end{aligned} \tag{2.2}$$

Observe that convex programming is a generalization of linear programming. This can be seen by letting $f(x) = c^\top x$ and $f_i(x) = A_i x - b_i$.

2.3.2. Non-Convex Non-linear Programming

When the function f or functions f_i are non-convex, this becomes a non-convex nonlinear programming problem. There are a few complexity issues with this.

IP AS NLP As seen above, quadratic constraints can be used to create a feasible region with discrete solutions. For example

$$x(1-x) = 0$$

has exactly two solutions: $x = 0, x = 1$. Thus, quadratic constraints can be used to model binary constraints.

Binary Integer programming (BIP) as a NLP:

NP-Hard

Given a matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$ and vector $c \in \mathbb{R}^n$, the *binary integer programming* problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & \cancel{x \in \{0,1\}^n} \\ & x_i(1-x_i) = 0 \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{2.3}$$

2.3.3. Machine Learning

Machine learning problems are often cast as continuous optimization problems, which involve adjusting parameters to minimize or maximize a particular objective. Frequently they are convex optimization problems, but many turn out to be nonconvex. Here are two examples of how these problems arise at a glance. We will see examples in greater detail later in the book.

Loss Function Minimization

In supervised learning, this objective is typically a loss function L that quantifies the discrepancy between the predictions of a model and the true data labels. The aim is to adjust the parameters θ of the model to minimize this loss. Mathematically, this can be represented as:

$$\min_{\theta} L(\theta) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} l(y_i, f(x_i; \theta)) \tag{2.4}$$

where N is the number of data points, l is a per-data-point loss (e.g., squared error for regression or cross-entropy for classification), y_i is the true label for the i-th data point, and $f(x_i; \theta)$ is the model's prediction for the i-th data point with parameters θ.

Clustering Formulation

Clustering, on the other hand, seeks to group or partition data points such that data points in the same group are more similar to each other than those in other groups. One popular method is the k-means clustering algorithm. The objective of k-means is to partition the data into k clusters by minimizing the within-cluster sum of squares (WCSS). The mathematical formulation can be given as:

$$\min_{\mathbf{c}_1,\ldots,\mathbf{c}_k} \sum_{j=1}^{k} \sum_{x_i \in C_j} \left\| x_i - \mathbf{c}_j \right\|^2 \tag{2.5}$$

where C_j represents the j-th cluster and $\mathbf{c}_j$ is the centroid of that cluster.

This encapsulation presents a glimpse into how ML problems are framed mathematically. In practice, numerous algorithms, constraints, and regularizations add complexity to these basic formulations.

2.4 Mixed-Integer Non-Linear Programming (MINLP)

2.4.1. Convex Mixed-Integer Non-Linear Programming

2.4.2. Non-Convex Mixed-Integer Non-Linear Programming

Part I

Linear Programming

3. MINLP

3.1 MINLP and Global Optimization

4. MINLP

Gurobi 9.0 webinar on non-convex quadratic programming

4.0.1. Relaxations and Convex Hulls

For a problem with a linear objective function

$$\min c^\top x \tag{4.1}$$
$$x \in \mathscr{X} \tag{4.2}$$

for some feasible set $\mathscr{X}$. Let $p^*_{\mathscr{X}}$ denote the optimal objective value.

Theorem 4.1: L

t $\mathscr{F} \subseteq \mathbb{R}^n$ (not necessarily convex) be compact and let $\mathscr{R} \supseteq \text{conv}(\mathscr{F})$. Then

$$p^*_{\mathscr{F}} = p^*_{\text{conv}(\mathscr{F})} \geq p^*_{\mathscr{R}},$$

that is,

1. *optimizing over the convex hull of feasible solutions returns the same objective value,*
2. *optimizing over a relaxation returns a lower bound.*

4.1 Extended Space and Spatial Branch and Bound

We will discuss spatial branch and bound in the context of liftings of nonconvex problems to an extended space. This means that we will reformulate the problem by adding new variables. Specifically, consider a quadratic optimization problem with linear constraints (where Q is symmetric)

$$\min \quad x^\top Q x + c^\top x \tag{4.1}$$
$$Ax \leq b \tag{4.2}$$
$$x \in \mathbb{R}^n \tag{4.3}$$

Let $\mathscr{F} = \{x \in \mathbb{R}^n : Ax \leq b\}$ be the feasible region of the problem.

4.1.0.1. Extended Space

For any product terms $x_i x_j$ in the formulation, construct a new variable Y_{ij} as

$$Y_{ij} = x_i x_j. \tag{4.4}$$

If all product terms are present, this can be written as

$$Y = xx^\top. \tag{4.5}$$

We will use this notation, although it is not always necessary to constructed a lifted variable when the corresponding products are not present.

Now the reformulation of the quadratic minimization problem can be written as

$$\begin{aligned} \min \quad & \tilde{Q} \circ Y + c^\top x && (4.6)\\ & Ax \leq b && (4.7)\\ & Y = xx^\top && (4.8)\\ & x \in \mathbb{R}^n && (4.9)\\ & Y \in \mathbb{R}^{n \times n} && (4.10) \end{aligned}$$

The objective function is now linear, which is very nice!

Now, let

$$\mathscr{F}_{\text{lift}} = \{(x,Y) \in \mathbb{R}^n \times \mathbb{R}^{n \times n} : Ax \leq b, Y = xx^\top\}.$$

The set $\mathscr{F}$ is non-convex, which is hard to deal with. So instead, we consider some convex relaxation $\mathscr{R}$ of $\text{conv}(\mathscr{F})$. We restrict that

$$\text{conv}(\mathscr{F}) \subseteq \mathscr{R} \subseteq \{(x,Y) : Ax \leq b\},$$

that is, we enforce the original constraints from $\mathscr{F}$ on the x variables. Thus, for any solution $(\bar{x}, \bar{Y}) \in \mathbb{R}$, although it may be that $(\bar{x}, \bar{Y}) \notin \mathscr{F}$, we do still have that $\bar{x}$ is feasible for the original problem.

$$\begin{aligned} \min \quad & \tilde{Q} \circ Y + c^\top x && (4.11)\\ & Ax \leq b && (4.12)\\ & (x,Y) \in \mathscr{R} && (4.13) \end{aligned}$$

4.1.1. Spatial Branch and Bound

The goal of spatial branch and bound is to sequentially identify subregions of the feasible set with lower bounds that are worse than the best known feasible solution, and hence you can *prune* (remove) that region from the space of places to look for an optimal solution.

In this algorithm, we build a *branch and bound tree* that starts with a root node and then branches on nodes as we go.

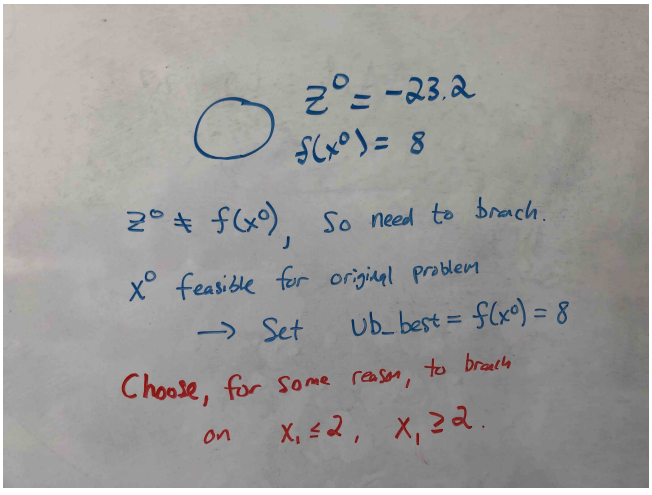
Process at a node:

Input: A feasible region $\mathscr{F}^i \subseteq \mathscr{F}$, a upper bound ubbest on $p^*_{\mathscr{F}}$

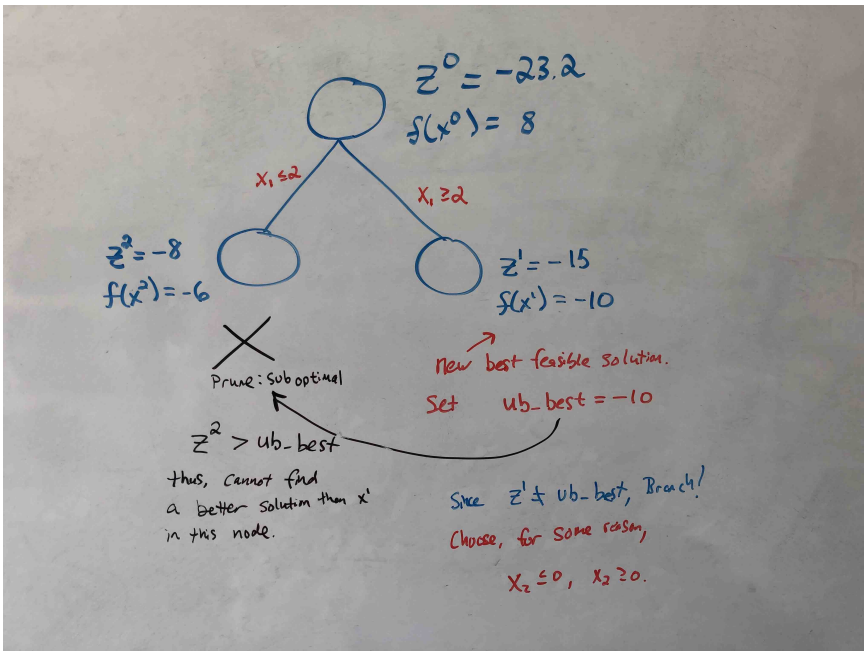
1. Write down the lifted problem (4.6).
2. Construct a relaxation $\mathscr{R}$ of $\mathscr{F}^i_{\text{lift}}$.
3. Optimize the relaxed lifted problem (4.11) and return an objective value z^i and a point $(x^i, Y^i) \in \mathscr{R}$. Note that
   - $x^i \in \mathscr{F}^i \subseteq \mathscr{F}$
   - $p^*_{\mathscr{F}} \leq p^*_{\mathscr{F}^i} \leq f(x^i)$
   - $p^*_{\mathscr{F}^i} \geq z^i$
4. If $f(x^i) <$ ubbest, set ubbest $\leftarrow f(x^i)$.
   (a) If $\mathscr{F}^i = \emptyset$, i.e., is infeasible, **prune this node.**
   (b) If $z^i >$ ubbest, **prune this node.** No optimal solution for $\mathscr{F}$ can be found in $\mathscr{F}^i$.
   (c) If $f(x^i) = z^i$, then $p^*_{\mathscr{F}_i} = f(x^i)$. Return x^i and **prune this node: found an optimal solution in** $\mathscr{F}^i$ (but this may not be optimal for $\mathscr{F}$).
   (d) Else, if $z^i < f(x^i)$ and $z_i <$ ubbest, then **Branch!**. Create two subproblems $\mathscr{F}^{i+1}$ and $\mathscr{F}^{i+2}$ such that $\mathscr{F}^i = \mathscr{F}^{i+1} \cup \mathscr{F}^{i+2}$, and process each node using this algorithm.

Here is an example of a branch and bound tree. It is important that each node ends in either infeasible, suboptimal, or optimal for the subproblem $\mathscr{F}^i$. Subproblems are created by changing lower or upper bounds on the variables x_i. The numbers in the example are fictitious and do not come from any particular function.

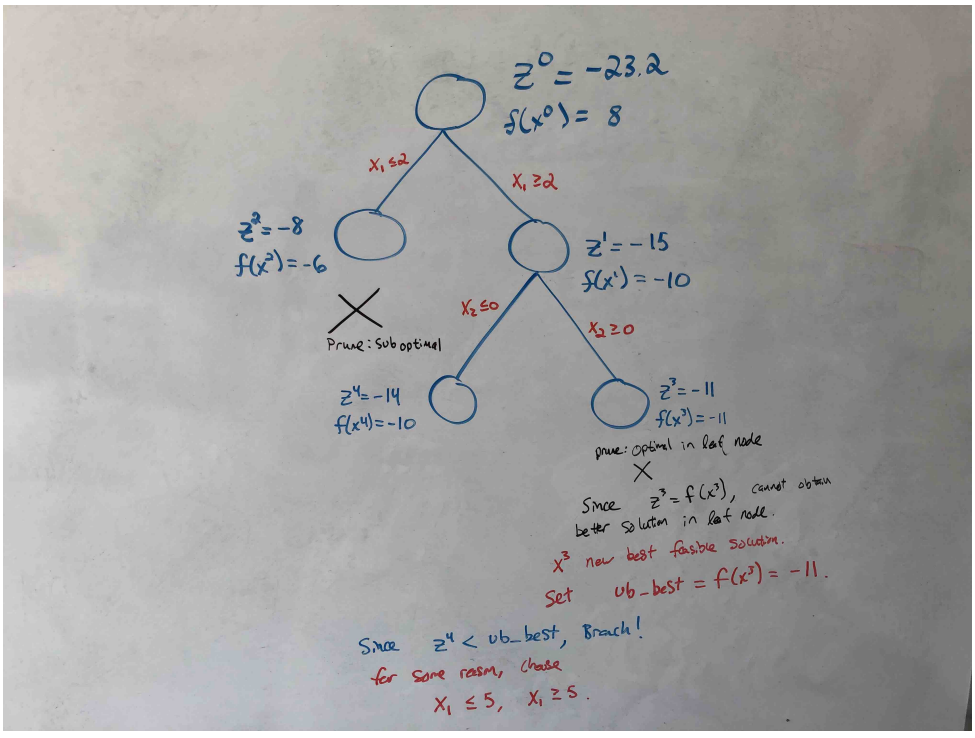
Step 0:



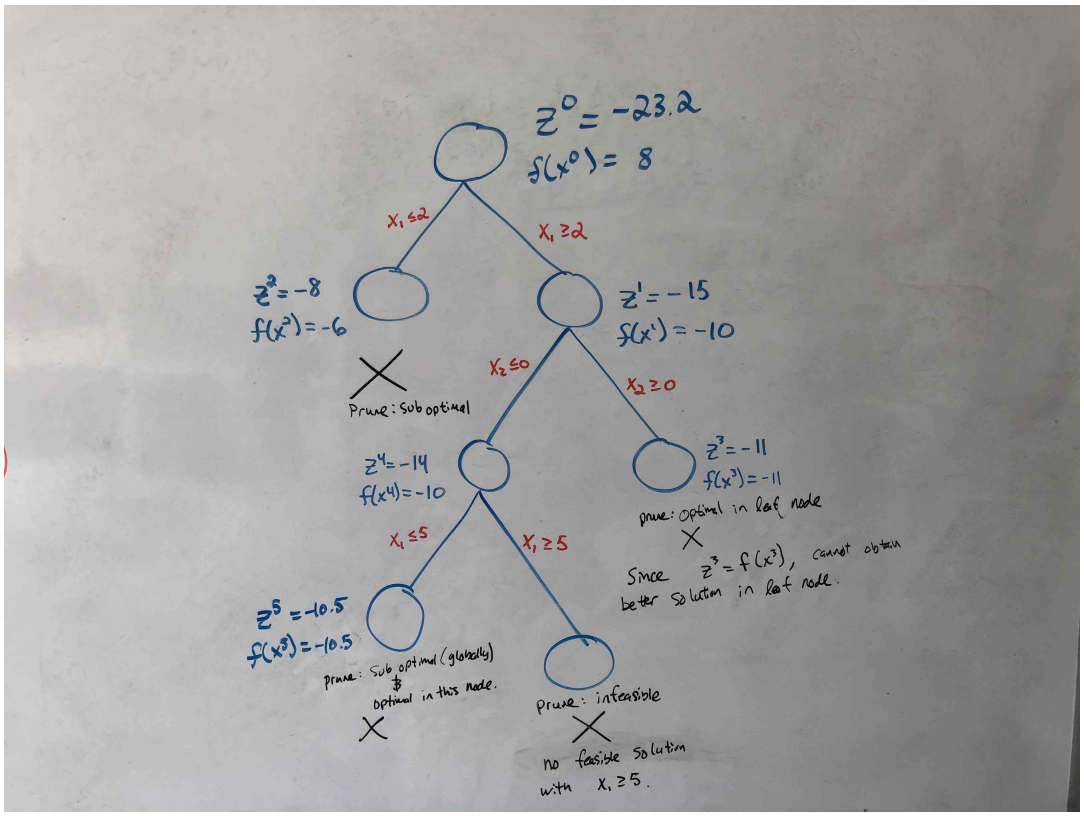
Step 1:


$z^0 = -23.2$
$f(x^0) = 8$
$x_1 \leq 2$
$x_1 \geq 2$
$z^2 = -8$
$f(x^2) = -6$
$z^1 = -15$
$f(x^1) = -10$
Prune: Sub optimal
new best feasible solution.
Set ub_best = −10
$z^2 > ub\_best$
thus, cannot find a better solution than x^1 in this node.
Since $z^1 \leq ub\_best$, Branch!
Choose, for some reason,
$x_2 \leq 0$, $x_2 \geq 0$.

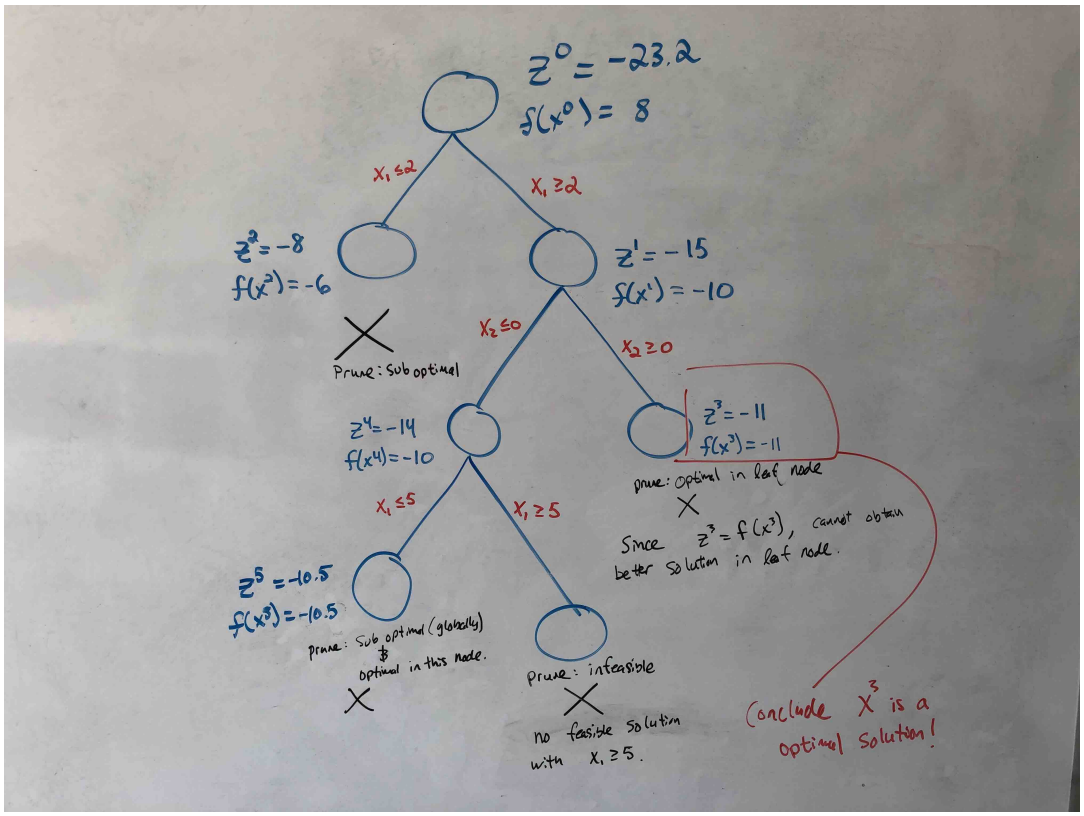
Step 2:


$z^0 = -23.2$
$f(x^0) = 8$
$x_1 \leq 2$
$x_1 \geq 2$
$z^2 = -8$
$f(x^2) = -6$
$z^1 = -15$
$f(x^1) = -10$
$x_2 \leq 0$
$x_2 \geq 0$
Prune: Sub optimal
$z^4 = -14$
$f(x^4) = -10$
$z^3 = -11$
$f(x^3) = -11$
prune: optimal in leaf node
Since $z^3 = f(x^3)$, cannot obtain better solution in leaf node.
x^3 new best feasible solution.
Set ub_best $= f(x^3) = -11$.
Since $z^4 < ub\_best$, Branch!
for some reason, choose
$x_1 \leq 5$, $x_1 \geq 5$.

Step 3:



Step 4:



There are several methods of construction relaxations $\mathbb{R}$ of $\mathscr{F}_{\text{lift}}$ including McCormick envelopes, using more inequalities from the Boolean Quadric Polytope, or SDP relaxations. These will be explored in the

next few subsections.

4.2 McCormick Envelope

For two variables, $l_i \leq x_i \leq u_i$ and $l_j \leq x_j \leq u_j$, we first write these inequalties as

$$x_i - l_i \geq 0, \quad u_i - x_i \geq 0 \tag{4.1}$$
$$x_j - l_j \geq 0, \quad u_j - x_j \geq 0 \tag{4.2}$$

Multiplying any two inequalties yields a quadtaric inequality, that can be linearized using the Y_{ij} variable. For example,

$$0 \leq (x_i - l_i)(x_j - l_j) = x_i x_j - l_i x_j - l_j x_i + l_i l_j = Y_{ij} - l_i x_j - l_j x_i + l_i l_j. \tag{4.3}$$

Enumerating all four pairs yields the McCormick Inequalities.

McCormick Inequalities:

$$l_i x_j + l_j x_i - l_i l_j \leq Y_{ij} \leq u_i x_j - l_j x_i + u_i l_j \tag{4.4}$$
$$u_i x_j + u_j x_i - u_i u_j \leq Y_{ij} \leq u_j x_i - l_i x_j + u_j l_i \tag{4.5}$$

4.3 Relaxation Linearization Technique (RLT)

The McCormic envelope inequalities are in fact a specific type of RLT inequality. In particular, for any two valid inequalities

$$a^1 x \leq b_1$$
$$a^2 x \leq b_2$$

the inequality

$$(a^1 x - b_1)(a^2 x - b_2) \geq 0$$

is also a valid inequality. Using the lifted variables this could be rewritten as

$$\Gamma \circ Y + \beta^\top x + \alpha \geq 0$$

which is a valid linear inequality for $\mathscr{F}_{\text{lift}}$. This inequality is referred to as an *RLT inequality*.

4.3.1. Boolean Quadric Polytope (BQP)

In this section, we assume that the upper and lower bounds are normalized, that is,

$$0 \leq x \leq 1.$$

This can be achieved easily by an affine transformation of the problem. Alternatively, one can extract generalizations of the results in this section to arbitrary lower and upper bounds via an affine transformation in the reverse direction.

Boolean Quadric Polytope (BQP):

$$\text{BQP} = \text{conv}\left((x,Y) \in \{0,1\}^{n+E} | Y_{ij} = x_i x_j \ \ \forall (i,j) \in E\right) \tag{4.1}$$

Theorem 4.2

$$\text{BQP} = \text{conv}\left((x,Y) \in [0,1]^{n+E} | Y_{ij} = x_i x_j \ \ \forall (i,j) \in E\right) \tag{4.2}$$

[Akshay-Gupte2019]

Triangle Inequalities:

$$\begin{aligned} x_i + x_j + x_k - y_{ij} - y_{ik} - y_{jk} &\leq 1 \\ -x_i + y_{ij} + y_{ik} - y_{jk} &\leq 0 \\ -x_j + y_{ij} - y_{ik} + y_{jk} &\leq 0 \\ -x_k - y_{ij} + y_{ik} + y_{jk} &\leq 0 \end{aligned}$$

Clique Inequalities:

$S \subseteq V$, $|S| \geq 3$, $1 \leq \alpha \leq |S| - 2$, $\alpha \in \mathbb{Z}$

$$\alpha x(S) - y(E(S)) \leq \frac{\alpha(\alpha+1)}{2}$$

? Inequalities:

4.3.2. SDP Relaxation

4.4 Lagrangian Relaxation

Consider a polynomial optimization problem

4.5 New Results

Theorem 4.3: Santana-Dey 2018

Let $S = \{x \in \mathbb{R}^n : x^\top Q x + cT^\top x = g, x \in P\}$ *for* $P = \{x : Ax \leq b\}$ *and* Q *symmetric. Then* $\text{conv}(S)$ *is SOCr. Proof is constructive.*

Theorem 4.4: Gupte et. al. 2019

If G *has nice structure, then* $\text{BQP}(G)$ *has a polynomial size extended formulation.*

4.6 Polynomial Optimization - Algebraic Techniques

4.7 Cylindrical Algebraic Decomposition

4.8 Gröbner Bases

4.9 Quantifier Elimination and Semi-algebraic Sets

Theorem 4.5

Any alternating quantifier set can be recast as a semi-algebraic set.

5. Approximations of SOCP and SDP

Some notes on applications for SDP: http://www.seas.ucla.edu/~vandenbe/publications/sdp-apps.pdf

5.1 Outer linear approximation of SDP

We describe a cutting plane technique for solving an SDP using Linear Programming and an oracle for finding eigenvectors of a matrix. This cutting plane technique could be applied to many other types of problems.

SDP:

$$\begin{aligned} \min \quad & c^\top x \\ \text{such that} \quad & Ax \leq b \\ & \sum_{i=1}^{n} F_i x_i + G \preceq 0 \qquad (*) \\ & x \in \mathbb{R}^n \end{aligned}$$

Let x^{LP} be the solution to the linear constraints to this problem, that is, we solve without the SDP constraint (**??**).

- If x^{LP} satisfies (**??**), then it is optimal.
- Otherwise, we find a new inequality $d^\top x \leq f$ that is valid for the problem but cuts off x^{LP}.

Let $A = \sum_{i=1}^{n} F_i x_i^{\text{LP}} + G$. If $A \not\succeq 0$, then we compute an eigenvector v of A with associated eigenvalue $\lambda > 0$.

Then,

$$v^\top \left(\sum_{i=1}^{n} F_i x_i + G \right) v \leq 0 \tag{5.1}$$

or equivalently

$$\sum_{i=1}^{n} v^\top F_i v x_i + v^\top G v \leq 0 \tag{5.2}$$

is a valid inequality for the problem that cuts off x^{LP}. Add this inequality to the LP and recompute x^{LP}. Continue this process until x^{LP} is feasible or close to feasible.

6. Discretization Approaches

6.1 Complete discretization - all binary variables

Let $x \in [l,u]$. Then we can replace x by an approximation in binary variables. Most common is to replace

$$x = l + \frac{1}{2^n}\sum_{i=1}^{m} 2^i \alpha_i \tag{6.1}$$

where $m = \lceil log_2(u-l) \rceil$. More generally, any integer base $b > 2$ can be used as

$$x = l + \varepsilon \sum_{i=1}^{n} b^i \left(\sum_{k=0}^{b} k\alpha_{ik} \right) \qquad \sum_{k=0}^{b} \alpha_{ik} = 1. \tag{6.2}$$

Note that for $b = 2$, we have variables $\alpha_{i0} + \alpha_{i1} = 1$, which reduces to the above case by eliminating the variable α_{i0}.

Under this transformation, the main problem is now a binary problem. Next, for any product appearing in the formilation,

$$\prod_{i \in I} \alpha_i$$

we can create a new variable $\beta_I = \prod_{i \in I} \alpha_i$. Since $\alpha_i \in \{0,1\}$, $\beta_I \in \{0,1\}$, and we can model their equivalence as

$$\beta_I \leq \alpha_i \quad \text{for all } i \in I \qquad \text{and } 1 + \sum_{i \in I} (\alpha_i - 1) \leq \beta_I. \tag{6.3}$$

Hence, the resulting formulation is a linear binary program in variables α_i and β_I.

6.2 Products of continuous variables - NMDT

We want to linearize the product of two variables using discretization. That is, given

$$\begin{aligned} x =& s \cdot y \\ s_{\min} \leq& s \leq s_{\max} \\ y_{\min} \leq& y \leq y_{\max} \end{aligned} \tag{6.1}$$

we want to convert this to a linear model by using binary variables. There are two main steps to this procedure. First, we discretize s into binary variables and then we model the product of each binary variable with the continuous variable y.

6.2.1. Discretizing a continuous variable + small error

Discretizing a continuous variable + small error:

Suppose that we have a variable z with bounds

$$z_{\min} \leq z \leq z_{\max}. \tag{6.2}$$

We can transform z via a variable w to the unit interval as

$$\begin{aligned} z &= (z_{\max} - z_{\min})w + z_{\min} \\ 0 \leq & w \leq 1 \end{aligned} \tag{6.3}$$

Next, we can convert w into a discrete variable $w_d \in \{\frac{i}{2^n} : i = 0, 1, \ldots, 2^n\}$ and an error $\Delta w \in [0, \frac{1}{2^n})$. This gives us

$$\begin{aligned} z &= (z_{\max} - z_{\min})(w_d + \Delta w) + z_{\min} \\ w_d &\in \{\tfrac{i}{2^n} : i = 0, 1, \ldots, 2^n\} \\ \Delta w &\in [0, \tfrac{1}{2^n}) \end{aligned} \tag{6.4}$$

Lastly, we convert the discrete w_d into binary variables as $w_d = \sum_{i=1}^{n} 2^{-i}\alpha_i$ with $\alpha_i \in \{0, 1\}$. Thus we have

$$\begin{aligned} z &= (z_{\max} - z_{\min}) \left(\sum_{i=1}^{n} 2^{-i}\alpha_i + \Delta w \right) + z_{\min} \\ \alpha_i &\in \{0, 1\} \text{ for } i = 1, \ldots, n \\ \Delta w &\in [0, \tfrac{1}{2^n}) \end{aligned} \tag{6.5}$$

6.2.2. Binary and continuous variables

Modeling: product of binary and continuous variable:

$$\begin{aligned} x &= \delta \cdot y \\ y_{\min} \leq & y \leq y_{\max} \\ \delta &\in \{0, 1\} \end{aligned} \tag{6.6}$$

Thus, we want to have the following happen

$$x = \begin{cases} y & \text{if } \delta = 1 \\ 0 & \text{if } \delta = 0 \end{cases} \tag{6.7}$$

This can be accomplished by

Modeling: product of binary and continuous variable - Reformulation:

$$\begin{aligned} \delta y_{\min} \leq & x \leq \delta y_{\max} \\ (1-\delta) y_{\min} \leq & \bar{x} \leq (1-\delta) y_{\max} \\ & y = x + \bar{x} \\ & \delta \in \{0,1\} \end{aligned} \tag{6.8}$$

Alternatively, we can eliminate the variable $\bar{x}$ by $\bar{x} = y - x$, leaving us with

Modeling: product of binary and continuous variable - Equivalent reformulation:

$$\begin{array}{rcrl} \delta y_{\min} & \leq & x & \leq \delta y_{\max} \\ (1-\delta) y_{\min} & \leq & y - x & \leq (1-\delta) y_{\max} \\ & & \delta & \in \{0,1\} \end{array} \tag{6.9}$$

6.2.3. References

[doi:10.1080/10556788.2016.1264397]

6.3 Polynomial Optimization

6.4 Polynomial Optimization

Polynomial Optimization

$$\begin{aligned} \max \quad & h(x) \\ \text{s.t.} \quad & f_1(x) = 0, \ \dots, \ f_m(x) = 0, \\ & g_1(x) \geq 0, \ \dots, \ g_{lc}(x) \geq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

Almost every combinatorial optimization can be modelled like this

Vertex Cover

Find a minimum size vertex set that covers every edge.

- $\min \ \sum x_i$
  $\text{s.t.} \quad x_i + x_j \geq 1 \qquad (i,j) \in E$
  $x_i^2 = x_i \qquad (x_i \in \{0,1\})$

  $x_i = \begin{cases} 1, \text{if } i \in V \\ 0 \text{ if } i \notin V \end{cases}$

OR.

- $\max \ \sum x_i$
  $\text{s.t.} \quad x_i \cdot x_j = 0$

  $x_i = \begin{cases} 0 \text{ if } i \in V \\ 1, \text{if } i \notin V. \end{cases}$

Binary Search

We instead solve the feasibility problem

$$\begin{aligned} & h(x) - \gamma \geq 0 \\ & f_i(x) = 0 \qquad i = 1, \dots, m \\ & g_i(x) \geq 0 \qquad i = 1, \dots, k \end{aligned}$$

When you know a range on values for objective function, we can do binary search with γ.

Goal: To understand the feasibility question with polynomial constraints.

Problem Setup: $\mathbb{K}$ = field, $\mathbb{K}[x_1, \dots, x_n] = \mathbb{K}[x] = R$

(ring of polynomials with coefficients in $\mathbb{K}$)

$x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ (monomial)

Solve $\begin{cases} f_i(x) = 0 & i = 1, \dots, m \\ g_i(x) \geq 0 & i = 1, \dots, k \end{cases}$

$p(x) = \sum_\alpha u_\alpha x^\alpha$, $\text{degree}(p(x)) = \max_{\alpha : u_\alpha \neq 0} \left(\sum_{i=1}^n \alpha_i \right)$

$F := \{ f_1, \dots, f_m \}$

$\langle F \rangle_R = \left\{ \sum_{i=1}^m \beta_i f_i \mid \beta_i \in R \right\}$ "Ideal generated by F"

Defn: A polynomial $s(x)$ is a Sum of Squares (SOS) if

$$s(x) = \sum_{i \in I} [g_i(x)]^2$$

Defn: Let $g_1, \dots, g_k \in R$

$$\text{cone}(G) = \left\{ s_0 + \sum_i s_i g_i + \sum_{\{i,j\}} s_{ij} g_i g_j + \dots + s_{ijk\dots} g_i g_j g_k \dots \right\}$$

s_α is a SOS polynomial $\}$

Lemma: Fredholm's Alternative

$\nexists$ x such that $Ax + b = 0 \iff \exists u$ such that $u^T A = 0$, $u^T b = 1$

Theorem Hilbert's (weak) Nullstellensatz

$\nexists$ x such that $f_i(x) = 0$, $i = 1, \dots, m$

$\iff \exists \beta_1, \dots, \beta_m \in R$ s.t. $\sum \beta_i f_i = 1$

$\iff 1 \in \langle F \rangle_R$

(Proof by Cox, Little, & O'Shea)

General Farkas' Lemma:

$\nexists\, x$ s.t. $Ax+b=0$, $Cx+d \geq 0$

$\iff \exists\, \mu, \lambda$ with $\lambda \geq 0$ $\nexists$

$\mu^T A + \lambda^T C = 0$

$\mu^T b + \lambda^T d = -1$

Proof uses: Hahn Banach Theorem = Separation Hyperplane Theorem

Theorem: Positive stellensatz (Stengle 1973)

$\nexists\, x$ such that $f_i(x)=0$, $i=1,\dots,m$

$\nexists\, g_i(x) \geq 0$, $i=1,\dots,k$

$\iff$

$\exists\ f \in \langle F \rangle_R$, $g \in \mathrm{Cone}(G)$ such that $f+g=1$

Example: $f_1(x) = x_1^2 - 1$, $f_2(x) = 2x_1x_2 + x_3$, $f_3(x) = x_1 + x_2$, $f_4(x) = x_1 + x_3$

$\{x \mid f_i(x) = 0\}$??

Algorithm to find infeasibility certificate:

$$\mu_1(x^2-1) + \mu_2(2x_1x_2+x_3) + \mu_3(x_1+x_2) + \mu_4(x_1+x_3) = 1$$

- Assuming μ_i's are constant

$$\Rightarrow \begin{cases} -\mu_1 = 1 \\ \mu_2 + \mu_4 = 0 \\ \mu_2 + \mu_4 = 0 \\ \mu_3 = 0 \\ \mu_1 = 0 \end{cases} \text{ infeasible} \Rightarrow$$

try with μ_i's as linear functions, then quadratics, etc.

Theorem: $\exists$ an exponential bound on the degrees of β_i's in the Hilbert infeasibility certificate.

Algorithm: NuLA (uses linear algebra to solve System of polynomial equations)

for $d = 1, \ldots, D$ (exponentially bounded)

1. try to find β_i of degree d s.t. $\sum \beta_i f_i = 1$
2. If yes, Stop → report infeasible
3. If no, continue with $d = d+1$.

If END → report feasible.

SOS $\quad p(x) = \sum_{i \in I} [q_i(x)]^2$

Theorem 1: $p(x) \in \mathbb{R}$ is SOS $\iff$ $p(x) = z^T Q z$ where Q is PSD, z is vector of monomials

Example: $p(x_1, x_2) = x_1^2 - x_1 x_2^2 + x_2^4 + 1$

$$= \frac{3}{4}(x_1 - x_2^2)^2 + \frac{1}{4}(x_1 + x_2^2) + 1$$

$$= \frac{1}{6}\begin{bmatrix} 1 & x_2 & x_2^2 & x_1 \end{bmatrix} \begin{bmatrix} 6 & 0 & -2 & 0 \\ 0 & 4 & 0 & 0 \\ -2 & 0 & 6 & -3 \\ 0 & 0 & -3 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \\ x_1 \end{bmatrix}$$

Defn: A symmetric matrix is PSD if $x^T A x \geq 0 \quad \forall x \in \mathbb{R}^n$

Theorem: Let A be a symmetric matrix.

A is PSD $\iff$ all eigenvalues of A are real & ≥ 0

$\iff$ $A = C^T C$ where C is a $k \times n$ matrix.

Proof Theorem 1: ($\Leftarrow$) $p(x) = z^T Q z = z^T C^T C z = \sum_i (C_i z)^2 = \text{SOS}$

($\Rightarrow$) $p(x) = \sum (q_i(x))^2 = x^T Q^T Q x$ where $Q = \begin{bmatrix} q_1 \\ \vdots \end{bmatrix}$ (q_i = coefficients, rows of C), $x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}$ all monomials.

□

6.4.1. Sums of Squares

Semi-Definite Programming (SDP)

$$\max \sum_{ij} C_{ij} X_{ij} = \text{Trace}(C \cdot X)$$

$$\text{s.t.} \quad \text{Trace}(A_i \cdot X) = b_i \qquad i \in I$$

$$X \succeq 0 \qquad (X \text{ is a } n \times n \text{ matrix})$$

Thm: Grötschel-Lovász-Schriver (GLS)

Optimization / test feasibility over a convex set $S \subseteq \mathbb{R}^d$ is equivalent to separation over S.

(i.e. we need a separation oracle).

Let $S = \{\Lambda \in \mathbb{R}^{n \times n} \mid \Lambda \succeq 0\} \subseteq \mathbb{R}^{n \times n}$

Separation oracle for S

- given a matrix Q, decide if Q belongs to S

  OR Find a separating hyperplane.

→ find all eigenvalues of Q

→ if $\lambda_i \geq 0 \ \forall i$, then $Q \in S$

→ otherwise $\exists\, y \in \mathbb{R}^n$ s.t.

$$Qy = \lambda y, \quad \lambda < 0$$

$$\Rightarrow y^T Q y = y^T \lambda y = \lambda \|y\|_2^2 < 0$$

→ Separating hyperplane $y^T \Lambda y = 0$

i.e. $y^T \Lambda y \geq 0$ is valid for S.

Therefore, GLS $\Rightarrow$ SDP can be solved (By Ellipsoid Method)

(but we can also do this with interior point methods)

Application in Combinatorial Optimization

Stable Set Problem (NP-Complete)

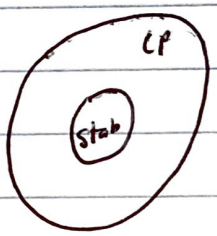
Find a maximum size stable set of vertices where no two vertices are connected by an edge

Prop: S is a stable set $\iff$ V/S is a vertex cover

$$\begin{aligned} \max \quad & \sum x_i \\ \text{s.t.} \quad & x_i + x_j \le 1 \qquad \forall (i,j) \in E \\ & x_i \in \{0,1\} \end{aligned}$$

$$LP(G) = \{ x \mid x_i + x_j \le 1 \quad \forall (i,j) \in E,\ 0 \le x_i \le 1 \}$$

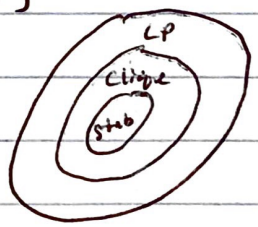
$$Stab(G) = Conv(\{ x \mid x_i + x_j \le 1,\ x_i \in \{0,1\} \})$$



Defn: A clique is a set of vertices s.t. $i,j \in E$ forall $i,j \in C$ (the clique)

Observation: For every clique C, at most 1 vertex from C belongs to a stable set.

$$Clique(G) = \{ x \mid \sum_{i \in C} x_i \le 1,\ 0 \le x_i \le 1 \}$$



Recall that we can write binary variables using quadratic constraints

- one formulation is

$$x_i^2 - x_i = 0$$
$$x_i \cdot x_j = 0 \qquad \forall (i,j) \in E$$

Application in Combinatorial Optimization

Stable Set Problem (NP-Complete)

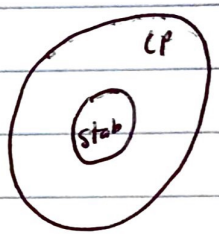
Find a maximum size stable set of vertices where no two vertices are connected by an edge

Prop: S is a stable set $\iff$ $V \setminus S$ is a vertex cover

$$\begin{aligned} \max \quad & \sum x_i \\ \text{s.t.} \quad & x_i + x_j \le 1 \qquad \forall (i,j) \in E \\ & x_i \in \{0,1\} \end{aligned}$$

$$LP(G) = \{ x \mid x_i + x_j \le 1 \quad \forall (i,j) \in E,\ 0 \le x_i \le 1 \}$$

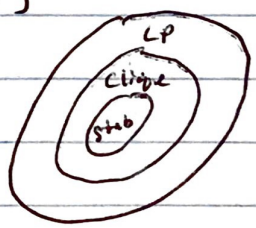
$$Stab(G) = Conv(\{ x \mid x_i + x_j \le 1,\ x_i \in \{0,1\} \})$$



Defn: A clique is a set of vertices s.t. $i,j \in E$ for all $i,j \in C$ (the clique)

Observation: For every clique C, at most 1 vertex from C belongs to a stable set.

$$Clique(G) = \{ x \mid \sum_{i \in C} x_i \le 1,\ 0 \le x_i \le 1 \}$$



Recall that we can write binary variables using quadratic constraints

- one formulation is

$$x_i^2 - x_i = 0$$
$$x_i \cdot x_j = 0 \qquad \forall (i,j) \in E$$

★ Introduce "linearizing variables" y_{ij}

$y_{ij} = x_i x_j$, $y_{jj} = x_j^2 = x_j$, $y_{0j} = y_{j0} = x_j$, $y_{00} = 1$

$\Rightarrow$ $Y = \begin{pmatrix} 1 \\ x \end{pmatrix} (1 \; x^T)$

★ Y is a positive semidefinite matrix

$\Rightarrow$ **Relaxation:** $Y \succeq 0$

i.e. need not be that $Y = (\;)(\;)$ for rank 1 matrices.

$\Rightarrow$ cannot recover x from the constraint.

$$TH(G) = \{x \mid Y \succeq 0,\ y_{ij} = 0 \ \forall (i,j) \in E,\ y_{0j} = y_{j0} = y_{jj} = x_j,\ y_{00} = 1,\ 0 \le x_j \le 1\}$$

"Theta body" from Lovasz

Theorem

$$Stab(G) \subseteq TH(G) \subseteq Clique(G)$$

proof: Clearly $Stab \subseteq TH(G)$ by construction.

WTS. $TH(G) \subseteq Clique(G)$.

Assume $x \in TH(G)$ $\Rightarrow$ $\exists\ Y \succeq 0$ in $TH(G)$ definition.

$\Rightarrow$ $v^T Y v \ge 0$ $\forall$ $v \in \mathbb{R}^{n+1}$

Take any clique C

set $v_i = \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$

Then $0 \le v^T Y v = \sum_{i,j} y_{ij} v_i v_j$

$= \sum_{i,j \in C} y_{ij} \underset{(-1)(-1)}{v_i v_j} + \sum_{i=1}^{n} y_{i0} \underset{1}{v_i v_0} + \sum_{j=1}^{n} y_{0j} \underset{1}{v_0 v_j} + y_{00} \underset{1}{v_0^2} \underset{1}{}$

(* $y_{ij} = 0$ if $(i,j) \in E$)

$= 0 + \sum_{j=1}^{n} y_{jj} - \sum_{i \in C} x_i - \sum_{i \in C} x_i + 1$

$\Rightarrow$ $\sum_{i \in C} x_i \le 1$ ☺ i.e. $x \in Clique(G)$ □

★ Introduce "linearizing variables" y_{ij}

$y_{ij} = x_i x_j$, $y_{jj} = x_j^2 = x_j$, $y_{0j} = y_{j0} = x_j$, $y_{00} = 1$

$\Rightarrow$ $Y = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 & x^T \end{pmatrix}$

★ Y is a positive semidefinite matrix

$\Rightarrow$ **Relaxation:** $Y \succeq 0$

i.e. need not be that $Y = (\ )(\ )$ for rank 1 matrices.

$\Rightarrow$ cannot recover x from the constraint.

$$TH(G) = \{x \mid Y \succeq 0,\ y_{ij} = 0\ \forall (i,j) \in E,\ y_{0j} = y_{j0} = y_{jj} = x_j,\ y_{00} = 1,\ 0 \le x_j \le 1\}$$

"Theta body" from Lovasz

Theorem

$$Stab(G) \subseteq TH(G) \subseteq Clique(G)$$

proof: Clearly $Stab \subseteq TH(G)$ by construction.

WTS. $TH(G) \subseteq Clique(G)$.

Assume $x \in TH(G) \Rightarrow \exists\ Y \succeq 0$ in $TH(G)$ definition.

$\Rightarrow v^T Y v \ge 0 \quad \forall\ v \in \mathbb{R}^{n+1}$

Take any clique C

set $v_i = \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$

Then $0 \le v^T Y v = \sum_{i,j} y_{ij} v_i v_j$

$= \sum_{i,j \in C} y_{ij} \underbrace{v_i v_j}_{(-1)(-1)} + \sum_{i=1}^{n} y_{i0} \underbrace{v_i v_0}_{1} + \sum_{j=1}^{n} y_{0j} \underbrace{v_0 v_j}_{1} + y_{00} \underbrace{v_0^2}_{1 \cdot 1}$

$= 0 + \sum_{j=1}^{n} y_{jj} - \sum_{i \in C} x_i - \sum_{i \in C} x_i + 1$

(★ $y_{ij} = 0$ if $(i,j) \in E$)

$\Rightarrow$ $\sum_{i \in C} x_i \le 1$ ☺ i.e. $x \in Clique(G)$ □

Max-Cut

Given a graph $G = (V, E)$ and weights on the edges, find a partition of $V = S \sqcup V/S$ such that the total weight of the edges crossing the partition is maximized.

Goemans-Williamson Approximate Algorithm (SDP)

There is an efficient (poly time) algorithm which gets within 82% of the optimal max cut.

- $\nexists$ any algorithm better than this unless $P = NP$

Formulation: $X_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases}$, $i \in V$.

$x_i \in \{\pm 1\} \iff x_i^2 = 1$

$x_i \neq x_j \iff \frac{1 - x_i x_j}{2} = 1$

$$\Rightarrow \max \sum w_{ij} \left(\frac{1 - x_i x_j}{2}\right)$$
$$\text{s.t. } x_i^2 = 1$$

Max-Cut

Given a graph $G = (V, E)$ and weights on the edges, find a partition of $V = S \sqcup V/S$ such that the total weight of the edges crossing the partition is maximized.

Goemans-Williamson Approximate Algorithm (SDP)

There is an efficient (polytime) algorithm which gets within 82% of the optimal max cut.

- $\nexists$ any algorithm better than this unless $P = NP$

Formulation: $x_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases}$, $i \in V$.

$x_i \in \{\pm 1\} \iff x_i^2 = 1$

$x_i \neq x_j \iff \frac{1 - x_i x_j}{2} = 1$

$$\Rightarrow \max \sum w_{ij} \left(\frac{1 - x_i x_j}{2}\right)$$

$$\text{s.t. } x_i^2 = 1$$

SOS Theory

Defn: f is convex if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$

$\forall\ x, y \in \mathbb{R}^n,\ \lambda \in [0,1]$

Prop: if f is twice-differentiable, then

$$f \text{ is convex} \iff H_f = \left[\left(\frac{\partial f}{\partial x_i \partial x_j}\right)\right] \succeq 0 \qquad \forall\ x \in \mathbb{R}^n$$

"the Hessian is PSD for all $x \in \mathbb{R}^n$"

Polynomial Matrices $P(x) = [P_{ij}]$ $(m \times m)$

(P_{ij}: polynomial in x)

Defn: $P(x)$ is a PSD matrix if $P(x)$ is PSD $\forall\ x \in \mathbb{R}^n$

Fact: $P(x)$ is PSD $\iff$ $y^T P(x) y \geq 0$ as an element of $\mathbb{R}[x, y]$, $y \in \mathbb{R}^m$

Defn: $P(x)$ is an SOS matrix if $\exists$ a polynomial matrix $M(x)$ such that $P(x) = M^T(x) M(x)$

Fact: $P(x)$ is SOS $\Rightarrow$ $P(x)$ is PSD

Fact: $P(x)$ is an SOS matrix $\iff$ $y^T P(x) y$ is a sum of squares polynomial.

Example PSD-Matrix not SOS-matrix

$$A(x) = \begin{bmatrix} x_1^2 + 2x_2^2 & -x_1x_2 & -x_1x_3 \\ -x_1x_2 & x_2^2 + 2x_3^2 & -x_2x_3 \\ -x_1x_3 & -x_2x_3 & x_3^2 + 2x_1^2 \end{bmatrix}$$

$A(x)$ is PSD, but $A(x)$ is not SOS

* $A(x)$ is not a Hessian matrix

since $\frac{\partial}{\partial x_3}\left(\frac{\partial^2 p}{\partial x_1^2}\right) \neq \frac{\partial}{\partial x_1}\left(\frac{\partial p^2}{\partial x_1 \partial x_3}\right)$

Note: if $n=1$, then PSD = SOS

Deciding Complexity of a polynomial function:

Defn: $P(x)$ is SOS-convex if $H(x)$ is an SOS-matrix

Clearly SOS-convex $\Rightarrow$ convex

Theorem (Amir Ali & Pablo Parrillo)

$\exists$ a polynomial that is convex, but not SOS-convex.

Note: Deciding $H(x) = \text{SOS}$ is easy since we just show that $y^T H(x) y$ is a SOS polynomial (use SDP to solve)

Time Complexity n = size of input

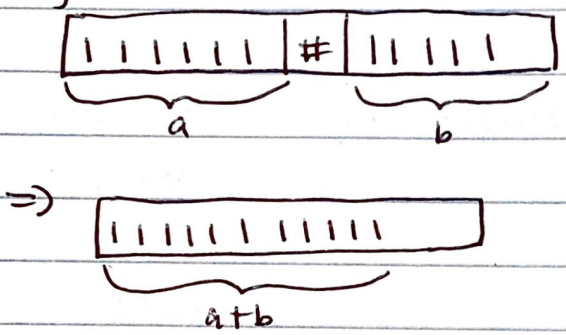
$T(n)$ = # of steps/transitions the turing machine makes

Space complexity

$S(n)$ = amount of storage required

Example: important to consider your input!!!

Adding: $a+b$

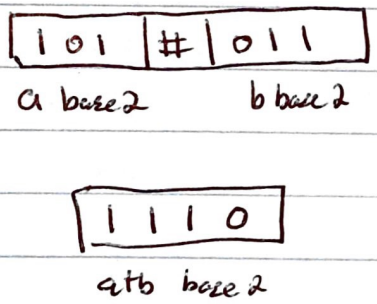


"unary encoding"

$T_1(n) = c_1 n$

length $a+b$.

Versus:



"binary encoding"

$T_2(n) = c_2 n$

only $\log(a) + \log(b)$ length

★ both linear time algorithms, but T_2 is exponentially better than T_1.

Defn: A problem is a set of strings on Σ

An instance is a particular string from the problem

Defn: The class of polynomial time problems $\mathcal{P}$ is the set of problems for which there exists a turing machine with polynomial time complexity.

Reduction: Problem B reduces to problem A if for all instances of B there exists a map

$R: B \to A$, $R(b) \in A$, s.t. R is polytime.

Defn: A and B are polynomially equivalent if A reduces to B and B reduces to A.

Example: Vertex Cover

- Optimization asks for the minimum vertex cover
- Feasibility asks for a vertex cover of size $\leq k$

Stable Set

- optimization asks for the maximum stable set
- feasibility asks for stable set of size $\geq n-k$

All of these are polynomially equivalent

Defn: A non-deterministic polynomial time (NP) turing machine has a map

$$f: Q \times \Sigma \to 2^Q \times \Sigma \cup \{\varepsilon\}$$

it can carry out parallel computations.

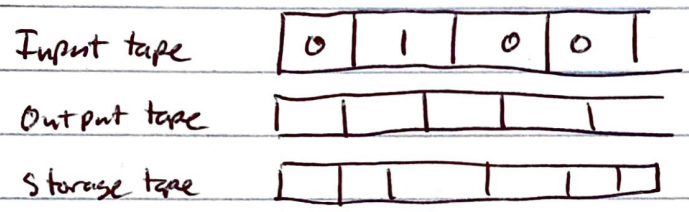
Alternative Defn: A problem is NP if there exists a polynomial time algorithm to check a given certificate.

Clearly $P \subseteq NP$. Probably $P \neq NP$.

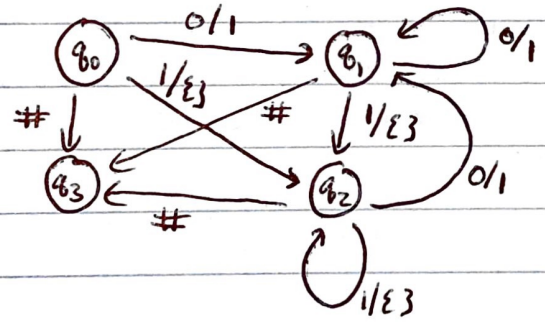
Complexity Theory

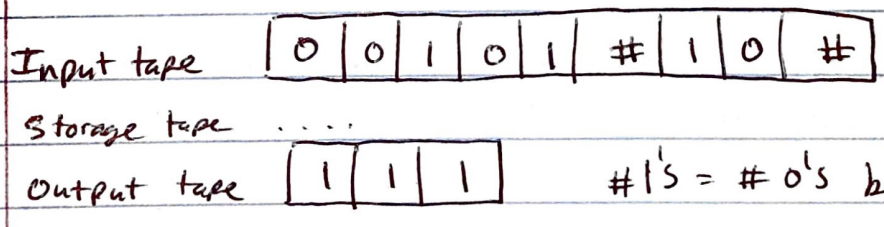
Defn: A Turing Machine includes the following

1. Q: a finite set of states $Q = \{q_0, \ldots, q_m\}$ (q_0: initial, q_m: final)
2. Σ "alphabet" - a set of symbols
3. f "transition function" $f: Q \times \Sigma \to Q \times \Sigma \cup \{e\}$ (e: empty symbol)
4. Tape : Input, Storage, & output



Example: Suppose $\Sigma = \{0, 1, \#\}$





#1's = # 0's before first #

"Any reasonable model of computation is equivalent to a turing machine"

Time Complexity n = size of input

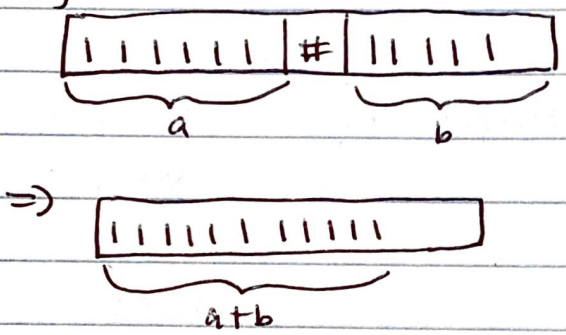
$T(n)$ = # of steps/transitions the turing machine makes

Space complexity

$S(n)$ = amount of storage required

Example: important to consider your input!!!

Adding: $a+b$

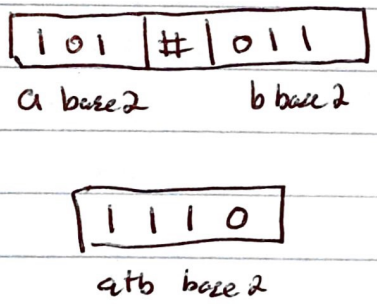


"unary encoding"

$T_1(n) = c_1 n$

length $a+b$.

Versus:



"binary encoding"

$T_2(n) = c_2 n$

only $\log(a) + \log(b)$ length

★ both linear time algorithms, but T_2 is exponentially better than T_1.

Time complexity n = size of input

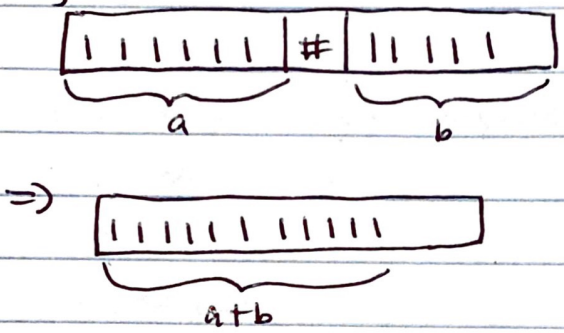
$T(n)$ = # of steps/transitions the turing machine makes

Space complexity

$S(n)$ = amount of storage required

Example: important to consider your input!!!

Adding: $a+b$

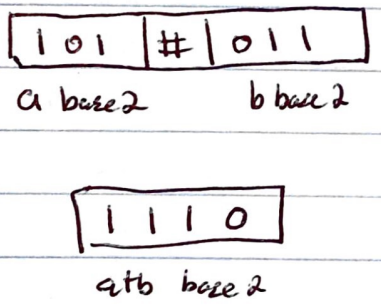


"unary encoding"

$T_1(n) = c_1 n$

length $a+b$.

versus:



"binary encoding"

$T_2(n) = c_2 n$

only $\log(a) + \log(b)$ length

* both linear time algorithms, but T_2 is exponentially better than T_1.

Defn: A problem is a set of strings on Σ

An instance is a particular string from the problem

Defn: The class of polynomial time problems $\mathcal{P}$ is the set of problems for which there exists a turing machine with polynomial time complexity.

Reduction: Problem B reduces to problem A if for all instances of B there exists a map

$R: B \to A$, $R(b) \in A$, s.t. R is polytime.

Defn: A and B are polynomially equivalent if A reduces to B and B reduces to A.

Example: Vertex Cover

- Optimization asks for the minimum vertex cover
- Feasibility asks for a vertex cover of size $\leq k$

Stable Set

- optimization asks for the maximum stable set
- feasibility asks for stable set of size $\geq n-k$

All of these are polynomially equivalent

Defn: A non-deterministic polynomial time (NP) turing machine has a map

$$f: Q \times \Sigma \to 2^Q \times \Sigma \cup \{\varepsilon\}$$

it can carry out parallel computations.

Alternative Defn: A problem is NP if there exists a polynomial time algorithm to check a given certificate.

Clearly $P \subseteq NP$. Probably $P \neq NP$.

Defn: A problem is a set of strings on Σ

An instance is a particular string from the problem

Defn: The class of polynomial time problems P is the set of problems for which there exists a turing machine with polynomial time complexity.

Reduction: Problem B reduces to problem A if for all instances of B there exists a map

$R: B \to A$, $R(b) \in A$, s.t. R is polytime.

Defn: A and B are polynomially equivalent if A reduces to B and B reduces to A.

Example: Vertex Cover

- Optimization asks for the minimum vertex cover
- Feasibility asks for a vertex cover of size $\leq k$

Stable Set

- optimization asks for the maximum stable set
- feasibility asks for stable set of size $\geq n-k$

All of these are polynomially equivalent

Defn: A non-deterministic polynomial time (NP) turing machine has a map

$$f: Q \times \Sigma \to 2^Q \times \Sigma \cup \{e\}$$

it can carry out parallel computations.

Alternative Defn: A problem is NP if there exists a polynomial time algorithm to check a given certificate.

Clearly $P \subseteq NP$. Probably $P \neq NP$.

Linear optimization

(P) $\max \{ c^T x \mid Ax = b, \ x \geq 0 \}$ $\quad A \in \mathbb{R}^{m \times n}$

(D) $\min \{ b^T y \mid A^T y \geq c \} = \min \{ b^T y \mid A^T y - s = c, \ s \geq 0 \}$

Theorem (Weak Duality)

If x is feasible for P, y feasible for D, then

$$b^T y \geq c^T x$$

Theorem (Strong Duality)

For each pair of linear programs, one of the following is true:

1. (P), (D) are both infeasible
2. (P) unbounded, (D) infeasible
3. (P) infeasible, (D) unbounded
4. (P) feasible, (D) feasible

$$\Rightarrow \quad b^T y^* = c^T x^*$$ ☺

Thm (Complementary Slackness)

When x^*, y^* are optimal solutions of P, D, then

$$x_i^* s_i^* = 0 \quad , i = 1, \ldots, n$$

Moral: Solving LP's results to solving the following system of equations and inequalities

$$Ax = b \quad , \quad A^T y - s = c \quad , \quad x^T s = 0$$

$$x \geq 0 \quad , \quad s \geq 0$$

Simplex Method (1947 George Dantzig)

Satisfy $Ax=b$, $A^Ty-S=c$, $x \geq 0$

Defn: X is a Basic Feasible Solution if $\exists$ a submatrix A_B of A with rank m (invertible) such that

$$x_j = 0 \text{ if } j \notin B$$

(therefore $A_B x_B = b$, $x_B = A_B^{-1} b$)

Lemma: If there exists an optimal solution of (P), then there exists an optimal BFS of (P).
(Assume bounded solution)

proof: Need 2 things:

1. Every BFS = an extreme point of (P) and vice versa
2. Every point in P can be written as a convex combination of extreme points + direction of unboundedness.

hence $x^* = d + \sum \alpha_i v_i$

- note that $c^T d = 0$ since otherwise we could scale in that direction

then $c^T x^* = \sum \alpha_i c^T v_i \leq \max c^T v_i \Rightarrow c^T x^* = c^T v_i$

□

Representation

$$b = Ax = A_B x_B + A_N x_N \Rightarrow x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

$$z = c^T x = c_B^T x_B + c_N^T x_N$$
$$= c_B^T A_B^{-1} b + (c_N^T - \underbrace{c_B^T A_B^{-1}}_{y} A_N) x_N$$

$$c_j - y a_j = s_j$$

Pivot Rule: Select index from those non-basic variables that correspond to

$$s_j < 0$$

i.e. $c_j - y^T a_j = -s_j > 0$.

Many choices for pivot rules!!!

Entering variable: Call a_j the column vector of the entering variable $x_j = 0$.

increase x_j slowly by $t > 0$

$$x_B^{new} = x_B^{old} - t(A_B^{-1} a_j)$$

⇒ eventually one other variable will vanish
→ this variable leaves the basis
→ OR no variable vanishes ⇒ unbounded direction.

How to find first BFS? $Ax = b$, $x \geq 0$, $b \geq 0$ (otherwise multiply by -1)

Set $\bar{A} = (A, I_m)$, $\bar{x} = (x, x_{n+1}, \ldots, x_{n+m})$

Solve $\bar{A}\bar{x} = b$, $\bar{x} \geq 0$, using $\bar{x}_0 = (0, b)$

i.e.

$$\max \; -(x_{n+1} + \ldots + x_{n+m})$$
$$\text{s.t.} \quad \bar{A}\bar{x} = b$$

if max $= 0$ ⇒ we found an initial BFS
if max < 0 ⇒ original problem is infeasible.

Open Problem: Can you find a pivot rule s.t. the number of pivot rules necessary to get an optimal solution is polynomial in n & m?

Can you bound the diameter of a graph in n & m?

Interior Point Methods (1984 Karmarker)

Theorem: For all $\lambda > 0$, ~~[illegible]~~ * has a unique real solution $(x^*(\lambda), y^*(\lambda), s^*(\lambda))$ s.t.

1.) $x^*(\lambda)$ is an optimal solution to

$$\max\left(c^T x + \sum_{i=1}^{n} \lambda \log(x_i) \mid Ax = b, x \geq 0\right)$$

2.) In the limit, the point $(x^*(0), y^*(0), s^*(0))$ is an optimal solution to (P).

Defn: The set of all real solutions of * as $\lambda \to 0$ is the Central Path

* $Ax = b$, $X_i S_i = \lambda$, $A^T y - S = c$, $x \geq 0$, $s \geq 0$

Steps of IPM: 1. Set $\lambda = 1$, solve *, find initial point (x, y, s) (or close enough point)

Given $\epsilon > 0$,

while $\lambda > \epsilon$ REPEAT

$$\lambda \to \left(1 - \frac{1}{2\sqrt{n}}\right)\lambda$$

$$(x, y, s) \to (x + \Delta x, y + \Delta y, s + \Delta s)$$

$$A(x + \Delta x) = b \qquad A^T(y + \Delta y) - (s + \Delta s) = c$$

$$(x_i + \Delta x_i)(s_i + \Delta s_i) = \lambda$$

$$s_i + \Delta s_i \geq 0 \qquad x_i + \Delta x_i \geq 0$$

Cheat: $A(\Delta x) = 0$, $A^T(\Delta y) - \Delta s = 0$

$$s_i \Delta x_i + x_i \Delta s_i = \hat{\lambda}$$

LP Applications

- game theory
- combinatorial optimization
  - max flow / min cut
  - help Integer Programs
  - Approximation Algorithms

game theory:

2-person, 0-Sum game

win for one person = loss for other person

"robbers" Player 2 \ player 1 "cops"	1	2	3
1	4	-10	-10
2	-8	5	-8
3	-12	-12	9

pay off for player 1 = -(Payoff for player 2)

$\Delta X = \{x \mid \sum x_i = 1\}$ $\Delta y = \{y \mid \sum y_i = 1\}$

Mixed strategy: Police patrol site i with probability x_i

robbers attack site i with probability y_i

$\sum x_i = 1$ $\sum y_i = 1$

Expected Pay off: $\sum_{i,j} a_{ij} x_j y_i = y^T A x$ ← Police want to maximize this

Police try to find $\bar{x} \in \Delta x$ s.t.

$$\bar{x} = \arg\max_{x \in \Delta x} \left(\min_{y \in \Delta y} y^T A x \right)$$

⇒ Police can guarantee a pay off of at least

$$\text{Payoff} \geq \max_x \left(\min_y y^T A x \right)$$

Similarly, let $\bar{y} = \arg\min_{y \in \Delta y} \left(\max_{x \in \Delta x} (y^T A x) \right)$

Then $\max_x \min_y y^T A x \leq \bar{y}^T A \bar{x} \leq \min_y \max_x y^T A x$

Thm (Von-Neuman)

$$\max_x \min_y y^T A x = \min_y \max_x y^T A x$$

Corollary: $\bar{x}, \bar{y}$ is an equilibrium point

i.e. $\bar{y}^T A x \leq \bar{y}^T A \bar{x}$

and $y^T A \bar{x} \geq \bar{y}^T A \bar{x}$

$\Rightarrow$ no one has incentive to move.

Pseudo-Proof of Theorem:

Consider $\max_{x \in \Delta_x} \min_{y \in \Delta_y} y^T A x$

If we fix a particular x^*, then $\min_y y^T A x^* = \min_i (Ax^*)_i$

("Put all chips in one bucket")

LP formulation:

(Primal)

$$v = \max z$$
$$\text{s.t.} \quad z \leq a_i x \qquad a_i = i^{th} \text{ row of } A, \text{ it runs of } A.$$
$$\sum x_i = 1$$
$$x_i \geq 0$$

$\Rightarrow$ (Dual)

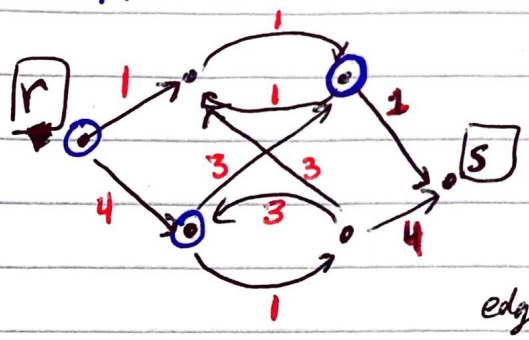
$$\min \gamma$$
$$\text{s.t.} \quad \gamma \geq y a^i$$
$$\sum y_i = 1$$
$$y \geq 0$$

Dual equivalent to

$$\min_y \max_x y^T A x.$$

Since $\text{opt(Primal)} = \text{opt(dual)}$ $\Rightarrow$ Result $\square$

Max-flow / min-cut Problem



Consider max flow from left to right

max flow $\geq$ min cut.

edges leaving ⊙ is 4

Theorem: Given a network/directed graph with vertices r, s, the maximum flow from r to s is <u>exactly</u> equal to the min cut between r & s.

Defn: A r-s cut is a partition of the vertices $V = R \sqcup S$ s.t. $r \in R$, $s \in S$, $S \cap R = \emptyset$

Defn: For a vertex v, $\delta^+(v)$ = set of all edges leaving v.
$\delta^-(v)$ = set of all edges entering v.

LP formulation:
- let x_e for all $e \in E$ be the amount of flow along the edge.
- for a vertex v, $x(\delta^+(v)) = \sum_{e \in \delta^+(v)} x_e$

and $x(\delta^-(v)) = \sum_{e \in \delta^-(v)} x_e$

Max flow

$$\max \quad x(\delta^+(r)) - x(\delta^-(r))$$

s.t.

$$x(\delta^+(v)) = x(\delta^-(v)) \quad \forall\, v \neq r, s$$

$$x_e \geq 0, \quad x_e \leq c_e$$

Min Cut

$$\min \sum_{e \in E} c_e y_e$$

s.t. $z_r = 1, \; z_s = 0$

$$z_w - z_v + y_{vw} \geq 0 \quad \forall\, e \in E$$

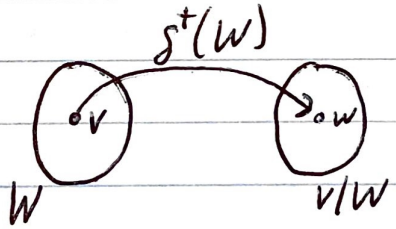
$$y_e \geq 0, \; z_v \in \mathbb{R}$$

Complementary Slackness

Let x^*, y^*, z^* be optimal solutions

then $y_e^*(c_e - x_e^*) = 0$, $(z_w - z_v + z_{vw}) \cdot x_{vw} = 0$

? $W = \{v \mid z_v^* \geq 1\}$, any edge in $\delta^+(W)$.

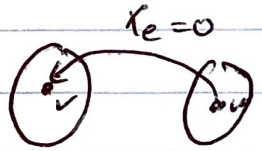


$z_v^* \geq 1, \; z_w^* < 1$

$(z_w^* - z_v^* + y_e^*) \geq 0 \rightarrow y_e^* > 0 \Rightarrow x_e^* = c_e$

"edge is at full capacity"

new case



for any edge in $e \in \delta^-(W)$, $e \in V/W$

$z_w^* - z_v^* + y_e^* \geq z_w^* - z_v^* > 0$

$z_v^* < 1, \; z_w^* \geq 1 \Rightarrow x_e = 0$

$\Rightarrow$ Capacity of the cut is equal to the flow value.

□

LP Solves IP

Theory of Totally Unimodular Matrices

Defn: A is called Totally Unimodular if every square submatrix has determinant $\pm 1, 0$.

Thm: $A = T.U.$ then

$$\min \{c^T x \mid Ax = b,\ x \geq 0\}$$

always has integral optimal solutions for every integral vector b.

Proof: Let x^* be the opt solution.

Then $\exists$ A_B $(m \times m)$ s.t. $x_B^* = A_B^{-1} b$, $x_N = 0$.

A_B^{-1} is integral.

$\Rightarrow$ by Cramers rule, x_B^* is integral.

Thm: Let A be a $\pm 1, 0$ matrix. If every column of A has at most one $+1$ and one -1, then A is T.U.

Proof: Let B be a submatrix of A.

if B has a 0 column (or row) then $\det(B) = 0$.

if B has a column w/ exactly 1, or -1, then expand about this column (induction)

if B has all 1's -1 columns, then sum of rows is 0.

$\Rightarrow$ linearly dependent $\Rightarrow \det(B) = 0$.

□

Prop: $A = \pm$ T.U. $\iff \begin{bmatrix} A \\ I \end{bmatrix}$ is T.U. $\iff \begin{bmatrix} A \\ I \end{bmatrix}$ is unimodular.

Vertex Cover:

- Find a subset of vertices which touch all edges.

Approximate Solution:

given weights c_v, $x_v \in \{0,1\}$

$$\min \sum_{v \in V} c_v x_v$$

$$\text{S.t.} \quad x_v + x_w \geq 1 \quad \forall \ (v,w) \in E$$

$$0 \leq x_v \leq 1 \quad \forall \ v \in V.$$

Solve using LP to get x^*.

- If $x^* \in \mathbb{Z}^n$, done!
- If not, let $\boxed{S = \{v \mid x_v^* \geq 1/2\}}$

Claim 1: S is a vertex cover

Claim 2: $\boxed{\text{weight}(S) \leq 2\,\text{weight}(S^*)}$

$$= 2 \sum_{v \in V} c_v x_v^*$$

□

Ellipsoid Method (1979) Kachiyan

input: a convex set $S \subseteq \mathbb{R}^n$
given to you by a separation oracle
& ϵ s.t. $Vol(S) > \epsilon$.

Output: A point $x \in S$ or "$S = \emptyset$."

Idea: Assume $\exists$ an ellipsoid

$$E_{M,z_0} = \{x \in \mathbb{R}^n \mid (x-z_0)^T M (x-z_0) \leq 1\}$$

M = positive definite

$S \subseteq E_{M,z_0}$, z_0 = center of E

* Check if $z_0 \in S$.
→ if $z_0 \notin S$, find new ellipsoid that encompasses $H_k^+ \cap E_k$.

Construct $E_{M_{k+1}, z_{k+1}}$ & Repeat while

$$Vol(E_{M_k z_k}) > vol(S).$$

Lemma: one can explicitly construct from $E_{M_k z_k}$ & H_k,
the exact equations for $E_{M_{k+1} z_{k+1}}$

with minimum volume &

$$Vol(E_{M_{k+1} z_{k+1}}) < Vol(E_{M_k, z_k}) \cdot e^{\frac{-1}{2(n+1)}}$$

After k steps

$$Vol(E_{M_k z_k}) < vol(E_{M_0, z_0})\, e^{\frac{-k}{2(n+1)}}$$

Note: $vol(S) < vol(E_{M_0 z_0})\, e^{-k/2(n+1)}$

$$\Rightarrow \quad k \leq 2(n+1) \ln\left(\frac{vol(E_{M_0, z_0})}{vol(S)}\right)$$

Also: If $P = \{y \mid A^T y \leq b\}$, $A = n \times n$ matrix,

if $P \neq \emptyset$, then it must contain a ball of radius $2^{-7\phi^3}$ where $\phi = O(m \cdot n \cdot \max(\log_2 |a_{ij}|, \log_2 |b_i|))$

History: 1989 (Renegar, Gonzaga, Roos [Vial])

find an optimal ~~so~~ Interior Point Method in $O(n^3 L)$

2000 (Deza, Terlaky) → best possible ever.

Simplex Counter example (exponential)

Klee-Minty Cube

$$0 \leq x_1 \leq 1, \quad \epsilon x_{k-1} \leq x_k \leq 1 - \epsilon x_{k-1}, \quad k = 2, \ldots, n$$

Other algorithms for LP:

1.) Criss-Cross Method
2.) Perceptron Algorithm
3.) Motzkin Relaxation Algorithm
4.) Ven pala Style Algorithms

Open: Prove any of these are polynomial

Curvature

Total curvature = length of the Gauss map

Curve $\gamma \rightarrow \frac{\dot{\gamma}}{\|\gamma\|}$ tangent vector

- Gauss map takes point on curve to tangent vector pointing on the Sphere

Open: $2\pi n \geq$ total curvature

Strongly Polynomial $= O(n^k)$ k constant

* Holy grail for LP *

Linear optimization

(P) $\max \{ c^T x \mid Ax = b, \ x \geq 0 \}$ $\quad A \in \mathbb{R}^{m \times n}$

(D) $\min \{ b^T y \mid A^T y \geq c \} = \min \{ b^T y \mid A^T y - s = c, \ s \geq 0 \}$

Theorem (Weak Duality)

If x is feasible for P, y feasible for D, then

$$b^T y \geq c^T x$$

Theorem (Strong Duality)

For each pair of linear programs, one of the following is true:

1. (P), (D) are both infeasible
2. (P) unbounded, (D) infeasible
3. (P) infeasible, (D) unbounded
4. (P) feasible, (D) feasible

$$\Rightarrow \quad b^T y^* = c^T x^*$$ ☺

Thm (Complementary Slackness)

When x^*, y^* are optimal solutions of P, D, then

$$x_i^* s_i^* = 0 \quad , i = 1, \ldots, n$$

Moral: Solving LP's results to solving the following system of equations and inequalities

$$Ax = b \quad , \quad A^T y - s = c \quad , \quad x^T s = 0$$

$$x \geq 0 \quad , \quad s \geq 0$$

Simplex Method (1947 George Dantzig)

Satisfy $Ax=b$, $A^Ty-S=c$, $x \geq 0$

Defn: X is a Basic Feasible Solution if $\exists$ a submatrix A_B of A with rank m (invertible) such that

$$x_j = 0 \text{ if } j \notin B$$

(therefore $A_B x_B = b$, $x_B = A_B^{-1} b$)

Lemma: If there exists an optimal solution of (P), then there exists an optimal BFS of (P).
(Assume bounded solution)

proof: Need 2 things:

1. Every BFS = an extreme point of (P) and vice versa
2. Every point in P can be written as a convex combination of extreme points + direction of unboundedness.

hence $x^* = d + \sum \alpha_i v_i$

- note that $c^T d = 0$ since otherwise we could scale in that direction

then $c^T x^* = \sum \alpha_i c^T v_i \leq \max c^T v_i \Rightarrow c^T x^* = c^T v_i$

□

Representation

$$b = Ax = A_B x_B + A_N x_N \Rightarrow x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

$$z = c^T x = c_B^T x_B + c_N^T x_N$$
$$= c_B^T A_B^{-1} b + (c_N^T - \underbrace{c_B^T A_B^{-1}}_{y} A_N) x_N$$

$$c_j - y a_j = s_j$$

Pivot Rule: Select index from those non-basic variables that correspond to

$$s_j < 0$$

i.e. $c_j - y^T a_j = -s_j > 0$.

Many choices for pivot rules!!!

Entering variable: Call a_j the column vector of the entering variable $x_j = 0$.

increase x_j slowly by $t > 0$

$$x_B^{new} = x_B^{old} - t(A_B^{-1} a_j)$$

⇒ eventually one other variable will vanish
→ this variable leaves the basis
→ OR no variable vanishes ⇒ unbounded direction.

How to find first BFS? $Ax = b$, $x \geq 0$, $b \geq 0$ (otherwise multiply by -1)

Set $\bar{A} = (A, I_m)$, $\bar{x} = (x, x_{n+1}, \ldots, x_{n+m})$

Solve $\bar{A}\bar{x} = b$, $\bar{x} \geq 0$, using $\bar{x}_0 = (0, b)$

i.e.

$$\max \; -(x_{n+1} + \ldots + x_{n+m})$$
$$\text{s.t.} \quad \bar{A}\bar{x} = b$$

if max $= 0$ ⇒ we found an initial BFS
if max < 0 ⇒ original problem is infeasible.

Open Problem: Can you find a pivot rule s.t. the number of pivot rules necessary to get an optimal solution is polynomial in n & m?

Can you bound the diameter of a graph in n & m?

Interior Point Methods (1984 Karmarker)

Theorem: For all $\lambda > 0$, ~~[illegible]~~ * has a unique real solution $(x^*(\lambda), y^*(\lambda), s^*(\lambda))$ s.t.

1.) $x^*(\lambda)$ is an optimal solution to

$$\max\left(c^T x + \sum_{i=1}^{n} \lambda \log(x_i) \mid Ax = b, x \geq 0\right)$$

2.) In the limit, the point $(x^*(0), y^*(0), s^*(0))$ is an optimal solution to (P).

Defn: The set of all real solutions of * as $\lambda \to 0$ is the Central Path

* $Ax = b$, $X_i S_i = \lambda$, $A^T y - S = c$, $x \geq 0$, $s \geq 0$

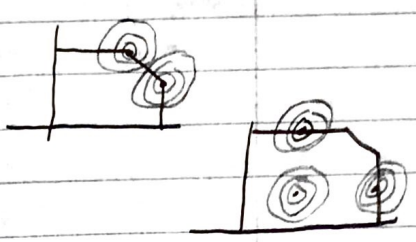
Intro NLP, Defn Concavity, Thm localmax => globalmax, $f''(x) \geq 0$ or ≤ 0

Non-Linear Programming

- solutions not necessarily at a corner point
- can have finitely many optimal solutions
- optimal solutions in the interior
- local optima are not necessarily global optima

Applications

* Portfolio Allocation
* Profit Maximization
* Volume Maximization



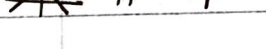
Suppose you have n investments. The return on investment i in the next period is R_i, a random variable. Let x_i be the percentage of money to invest in i.

$$\text{Max } Z = \underbrace{E\left[\sum_{i=1}^{n} x_i R_i\right]}_{\text{return}} - \underbrace{\mu}_{\text{indicates level of risk willingness}} \text{Var}\left[\sum_{i=1}^{n} x_i R_i\right]$$

Constraints: $\sum_{i=1}^{n} x_i = 1$, $x_i \geq 0 \ \forall i$ $\Rightarrow Z = \sum_{i=1}^{n} x_i E(R_i) - \mu \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \text{cov}(R_i, R_j)$ ← non-linear!

Solve $f'(x) = 0$. Is this sufficient and necessary for x to be a max? NO!

→ not sufficient: min, plateau, local max, local min

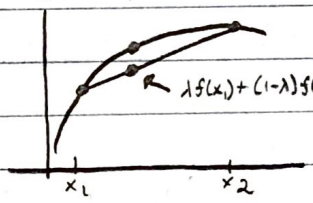
→ not necessary! not differentiable, boundary max or min, increasing function 

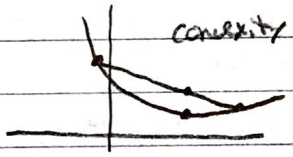
Want f continuous, differentiable, domain of f is unbounded, concave!

$\Rightarrow$ $f'(x_0) = 0$ iff x_0 is a global max.

Defn: A function is concave if for any x_1, x_2 in the domain of f and $\lambda \in [0,1]$ we have

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2)$$





Defn: A function is convex if " $\leq$ "

Theorem: If f is concave on $\mathbb{R}$, then any local maximum of f is a global max of f.

Proof: Let $\bar{x}$ be a local max of f. Suppose $\bar{x}$ is not a global max. Then there exists an $\bar{x}' \neq \bar{x}$ s.t. $\bar{x}'$ is a global max of f and $f(\bar{x}') > f(\bar{x})$. Thus, for $\lambda \in [0,1]$

$f(\lambda\bar{x} + (1-\lambda)\bar{x}') \geq \lambda f(\bar{x}) + (1-\lambda) f(\bar{x}') > \lambda f(\bar{x}) + (1-\lambda) f(\bar{x}) = f(\bar{x})$

Since we can choose λ arbitrarily close to 1, we can find find points with larger function values which violates that $\bar{x}$ is a local max. ↯ Thus $\bar{x}$ is a global max ■

Thm: concavity => continuity. no proof given.

Thm: If f'' exists everywhere then $f''(x) \geq 0$ everywhere $\Leftrightarrow$ f is convex

$f''(x) \leq 0$ everywhere $\Leftrightarrow$ f is concave

Solving $f'(x)=0$. Use numerical approximations because solving analytically can be a pain.

→ **Bisection Method:** (f is concave) Looking for global max x^* know lower & upper bound $\underline{x}$ and $\bar{x}$.

Let $\hat{x} = \frac{\underline{x}+\bar{x}}{2}$. If $f'(\hat{x})>0$ then $\underline{x} := \hat{x}$, If $f'(\hat{x})<0$ then $\bar{x} := \hat{x}$.

Stop when $\hat{x}$ is within ϵ of the optimal solution (pick ϵ in advance) $\Leftrightarrow$ $\bar{x} - \underline{x} < 2\epsilon$

* See handout for example with $f(x) = e^x \cos(x) + x$, Max on $[0, \pi]$.

Bisection method takes K iterations where $\frac{\Delta \bar{x}}{2^k} \leq 2\epsilon \Rightarrow K = \log_2\left(\frac{\Delta \bar{x}}{\epsilon}\right) - 1$

→ **Newton's Method:** root finding method. Converges very quickly to optimal solutions.

Given x_i, The equation of the tangent line $(x-x_i)m = (y-y_i)$

$\Rightarrow y = m(x-x_i) + y_i$. $y_i = f(x_i)$, $y=0$, $m = f'(x_i)$

$\Rightarrow 0 = f'(x_i)(x-x_i) + f(x_i)$ $\Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f(x_{i+1})}$

But, we want $f'(x)=0$, so $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$

→ **Screwing with Newtons Method**

loop!

Multivariate NLP

① All partial derivatives $=0$ OR ② on a boundary OR ③ where at least one partial derivative DNE

$\nabla f \equiv$ grad f is the vector of partial derivatives $\nabla f(x,y) = f_x(x,y)\hat{i} + f_y(x,y)\hat{j}$

① $\Leftrightarrow$ $\nabla f(x,y) = \vec{0}$

∇f points in the direction of maximum increase for f.

Directional Derivative: $f_{\vec{u}}(x,y) = \nabla f(x,y) \cdot \vec{u}$ ($\vec{u}$ is a unit vector)

$\vec{u}\cdot\vec{v} = \|\vec{u}\|\,\|\vec{v}\| \cos\theta$ where θ is the angle between $\vec{u}$ and $\vec{v}$.

$f_{\vec{u}}(x,y) = \|\nabla f(x,y)\|\,\|\vec{u}\| \cos\theta = \|\nabla f(x,y)\| \cos\theta$ what value of θ maximizes this? $\theta = 0$!

Hessian matrix $H(f) = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$ look at $\det(H)$

If $\nabla f(a,b) = \vec{0}$, ⋕3 $\det(H(a,b)) = 0$ Then second partials test fails

⋕⋕ $\det(H(a,b)) < 0$ you have a saddle point at (a,b)

⋕⋕ $\det(H(a,b)) > 0$ ⋕⋕ f_{xx} or $f_{yy} > 0$ $\Rightarrow$ local min at (a,b)

⋕⋕ f_{xx} or $f_{yy} < 0$ $\Rightarrow$ local max at (a,b)

$d(a,b) = f_{xx}(a,b)\, f_{yy}(a,b) - [f_{xy}(a,b)]^2$

If f_{xx} & f_{yy} are both positive why don't we necessarily have a min at (a,b)? Because of twisting

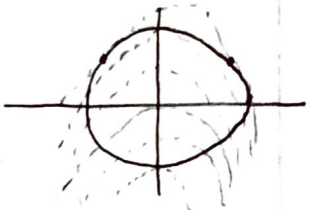
$f(x,y) = x^4 + y^4 - 4xy + 1$: $\nabla f(x,y) = (4x^3 - 4y)\hat{i} + (4y^3 - 4x)\hat{j} \Rightarrow \nabla f(x,y) = 0 \Leftrightarrow \begin{matrix} 4x^3 - 4y = 0 & \Leftrightarrow & x^3 = y \\ 4y^3 - 4x = 0 & \Leftrightarrow & y^3 = x \end{matrix}$

$x = 0, \pm 1$ If $x=0$ $y=0$, if $x=1$, $y=1$, if $x=-1$, $y=-1$ next look at Hessian.

$\det(H)$ $(0,0)$ < 0 $\Rightarrow$ saddle

$(1,1)$ > 0, $f_{xx} > 0$ local min

$(-1,-1)$ > 0 $f_{xx} > 0$ local min



Lagrange Multipliers: Multivariate constrained optimization

Maximize $f(x,y) = x^2+y$ s.t. $x^2+y^2 = 1$

∇f is perpendicular to circle at optimal point. $g(x,y) = x^2+y^2$ when $g=1$. Think of circle as a level curve

∇g at optimal points perpendicular to the circle $\Rightarrow$ $\boxed{\nabla f(x,y) = \lambda \nabla g(x,y)}$ λ = Lagrange Multiplier

$\nabla f(x,y) = \langle 2x, 1\rangle$ $\nabla g(x,y) = \langle 2x, 2y\rangle$

$2x = \lambda 2x$ $\quad 1 = \lambda 2y$ $\quad x^2+y^2 = 1$ $\Rightarrow (0,1), (0,-1), \left(\pm\frac{\sqrt{3}}{2}, \frac{1}{2}\right)$

First method to produce max & min on interior. Use Lagrange Multipliers to find max/min on boundaries.

What does λ mean? - Let (x_0, y_0) be the optimal solution to maximize $f(x,y)$ s.t. $g(x,y) = c$.

express as $(x_0(c), y_0(c))$. How does $f(x_0(c), y_0(c))$ change as c changes?

$\frac{df}{dc}\Big|_{(x_0,y_0)} = \frac{\partial f}{\partial x_0}\frac{dx_0}{dc} + \frac{\partial f}{\partial y_0}\frac{dy_0}{dc}$ $\quad \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial y_0}\right) = \lambda\left(\frac{\partial g}{\partial x_0}, \frac{\partial g}{\partial y_0}\right)$

at a point of optimal value. $\Rightarrow \frac{\partial f}{\partial c}\Big|_{(x_0,y_0)} = \lambda\frac{\partial g}{\partial x_0}\frac{dx_0}{dc} + \lambda\frac{\partial g}{\partial y_0}\frac{dy_0}{dc} = \lambda\frac{\partial g}{\partial c}\Big|_{(x_0,y_0)} = \lambda \rightarrow$ shadow price

λ is the shadow price of f with respect to the constraint $g(x,y) = c$ [note: only instantaneous shadow price]

$f''(x) \le 0 \Leftrightarrow f$ is concave (if f'' exists $\forall x \in \mathbb{R}$)

Max $f(x,y)$
s.t. $g(x,y) = c$
$h(x,y) = d$
$\Big\}$ $\nabla f = \lambda \nabla g + \mu \nabla h$

Characterization of Concavity

Defn: A function $f(\vec{x})$ is concave, if $\forall \vec{x}_1, \vec{x}_2 \in \mathbb{R}^n$

$f(\lambda \vec{x}_1 + (1-\lambda)\vec{x}_2) \ge \lambda f(\vec{x}_1) + (1-\lambda) f(\vec{x}_2)$

Alternate characterization: Suppose f has continuous first partials in $\mathbb{R}^n$, then

f is concave $\Leftrightarrow$ $f(\vec{x}) \le f(\vec{x}_0) + \nabla f(\vec{x}_0)(\vec{x} - \vec{x}_0)$ $\forall \vec{x}, \vec{x}_0 \in \mathbb{R}^n$

$\rightarrow$ Single variable $y = f(x_0) + f'(x_0)(x - x_0)$

Taylor's Theorem in 2 variables (up to first order)

If f is twice-differentiable on $\mathbb{R}^2$ then for some (a,b) in the line segment between (x,y) and (x_0,y_0)

$f(x,y) = f(x_0,y_0) + f_x(x_0,y_0)(x-x_0) + f_y(x_0,y_0)(y-y_0) + \frac{1}{2}\left[f_{xx}(a,b)(x-x_0)^2 + 2f_{xy}(a,b)(x-x_0)(y-y_0) + f_{yy}(a,b)(y-y_0)^2\right]$

Using ∇f and matrices this is $f(x,y) = f(x_0,y_0) + \nabla f(x_0,y_0)\cdot\begin{bmatrix} x-x_0, & y-y_0\end{bmatrix} + \frac{1}{2}\begin{bmatrix} x-x_0, & y-y_0\end{bmatrix}\begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy}\end{bmatrix}\begin{bmatrix} x-x_0 \\ y-y_0\end{bmatrix}$

for $S: \mathbb{R}^n \to \mathbb{R}$, this is $f(\vec{x}) = f(\vec{x}_0) + \nabla f(\vec{x}_0)[\vec{x}-\vec{x}_0] + \frac{1}{2}[\vec{x}-\vec{x}_0]^T H(\vec{z})[\vec{x}-\vec{x}_0]$ for some $\vec{z}$

Note: $\vec{x}-\vec{x}_0$ is a column vector that is a convex combination of $\vec{x}$ and $\vec{x}_0$

★ **Theorem:** Suppose f has continuous second partial derivatives in $\mathbb{R}^n$. Then f is concave $\Leftrightarrow$ its Hessian Matrix is negative semi-definite.

Defn: A is negative semi-definite $\Leftrightarrow$ $\vec{x}^T A \vec{x} \le 0$ $\forall \vec{x} \in \mathbb{R}^n$ (or $\mathbb{C}^n$).

Proof: H is negative semidefinite $\forall \vec{z} \in \mathbb{R}^n \Leftrightarrow [\vec{x}-\vec{x}_0]^T H(\vec{z})[\vec{x}-\vec{x}_0] \le 0$ $\forall \vec{x}, \vec{x}_0 \in \mathbb{R}^n$

$\Leftrightarrow$ $f(\vec{x}) \le f(\vec{x}_0) + \nabla f(\vec{x}_0)[\vec{x}-\vec{x}_0]$ $\forall \vec{x}, \vec{x}_0 \in \mathbb{R}^n$

$\Leftrightarrow$ f is concave on $\mathbb{R}^n$ ■

Ex: $f(x_1,x_2,x_3) = 4x_1^2 + x_2^2 + x_3^2 - 2x_2x_3$ is convex. (Positive semi-definite Hessian)

details
↓

$$[x_1\ x_2\ x_3]\begin{bmatrix} 8 & 0 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 8x_1^2 + 2x_2^2 + 2x_3^2 - 4x_2x_3$$

$$= 8x_1^2 + 2(x_2^2 - 2x_2x_3 + x_3^2) = 8x_1^2 + 2(x_2 - x_3)^2 \Rightarrow \text{PSD Hessian} \Leftrightarrow \text{convex!}$$

Other characteristics of NSD

① All eigenvalues are non-positive (if)

② See text on Matrix Algebra for characterization terms of principle minors.

Thm: If $f_i(\vec{x})$ is concave, then $\sum_{i=1}^{n} f_i(\vec{x})$ is also

Thm: If $f(\vec{x})$ is concave, and $q > 0$ then

$q f(\vec{x})$ is concave AND $-q f(\vec{x})$ is convex.

Thm: If $f(\vec{x})$ is concave and $g(y)$ is increasing then $g(f(\vec{x}))$ is concave

Ex: $f(x_1, x_2, x_3) = e^{-(x_1^2 + x_2^2 + x_3^2)}$

e^x is increasing, $-x_1^2 - x_2^2 - x_3^2$ is concave

thus f is concave. Also

$$H_f(x,y) = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix} \Rightarrow \text{NSD} \Rightarrow \text{concave.}$$

6.4.2. Nullstellensatz

6.4.3. Positivestellensatz

6.4.4. Putinar's Positivestellensatz

6.4.5. Handelman Decompositions

6.5 Algebraic and Geometric Ideas

7. Algebraic and Geometric Ideas

Here we will discuss one more important application of Gröbner bases. We've seen how to solve polynomial equations, and now we're going to use Gröbner bases for integer programming. We consider the problem

$$\begin{array}{ll} \min & c \cdot x \\ \text{subject to} & Ax = b \\ & x \geq 0, x \in \mathbb{Z}^n \end{array}$$

where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$.

Roughly speaking, the first goal is to understand the vectors in the kernel of A that are integer. That is to say, we want to analyze the set $\ker(A) \cap \mathbb{Z}^n = \{x \mid Ax = 0, x \in \mathbb{Z}^n\}$. For that purpose, we define the main object here, the *toric map* or *toric ideal*.

Definition 7.1: T

e map $\pi_A : \mathbb{N}^n \to \mathbb{Z}^m$ is defined by $u \mapsto Au$.

We will be interested in the following extension $\widehat{\pi_A}$ of the map π_A. The map is defined

$$\widehat{\pi_A} : K[x_1, \ldots, x_n] \to K[t_1^{\pm}, \ldots, t_m^{\pm}].$$

Every variable may appear with a positive or negative exponent. It's not the usual way to think of rings of polynomials. Here we have Laurent polynomials. For instance, we may have

$$\frac{t_1}{t_2^2} - \frac{7t^2}{t_1^7} - 3t_1 t_2^7.$$

Laurent polynomials can be thought of as regular polynomials that are using variables of the form t_1 and $1/t_1$. We can now define a map by simply showing what happens to each variable x_i. The map $\widehat{\pi_A}$ maps

$$x_i \mapsto t_1^{a_{i_1}} t_2^{a_{i_2}} \cdots t_m^{a_{i_m}}$$

Here, a_i denotes the ith column of A. Suppose you have an arbitrary polynomial $f = \sum c_\alpha x^\alpha$. The map $\widehat{\pi_A}$ is linear, so

$$\widehat{\pi_A}(\sum c_\alpha x^\alpha) = \sum c_\alpha \widehat{\pi_A}(x^\alpha) = \sum c_\alpha (\widehat{\pi_A}(x_1)^{\alpha_1} \widehat{\pi_A}(x_2)^{\alpha_2} \cdots \widehat{\pi_A}(x_n)^{\alpha_n}).$$

Example 7.2: L

t A be the matrix

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix}.$$

Note that we could also write an example with some negative entries. Here the map is defined with spaces

$$K[x_1,\ldots,x_4] \to K[t_1,t_2].$$

Suppose that the polynomial f is $7x_1^3 + x_2x_3 - 7x_4^3$. What is $\widehat{\pi_A}(x_1)$? It's just t_1. What is $\widehat{\pi_A}(x_2)$? It's just t_1t_2, by looking at the second column of A. What is $\widehat{\pi_A}(x_3)$? It's just $t_1t_2^2$. Finally, $\widehat{\pi_A}(x_4)$ is $t_1t_2^3$. So when applied to f, we obtain $7t_1^3 + (t_1t_2)(t_1t_2^2) - 7(t_1t_2^3)^3$ which simplifies to $7t_1^3 + t_1^2t_2^3 - 7t_1^3t_2^9$.

This map is going to tell us everything we want to know about the elements of the kernel of A that are integer. Once we understand the kernel of A, if we somehow obtain a single feasible solution, we can use the kernel of A as candidates for feasible directions to move in. Much like in the simplex method, we will pivot into feasible directions and attempt to find a more optimal value. The directions are found by the strange map. We will look at the kernel of the map $\widehat{\pi_A}$.

The question is, what is $\ker(\widehat{\pi_A})$? That is, which polynomials map to zero under the projection $\widehat{\pi_A}$? The first fact we need to see is that this kernel is an ideal.

Lemma 7.3

The set $\ker(\widehat{\pi_A})$ is an ideal inside the polynomial ring $K[x_1,\ldots,x_n]$.

This ideal is called the *toric ideal*, which we will denote from now on by I_A. (The name of this ideal is unfortunately not suggestive/descriptive. There are mathematical reasons why these are called *toric*. The name comes from a torus action.) By the Hilbert Basis Theorem, there must be finitely many polynomials that generate this ideal. Our job is to compute these polynomials. Here is an example: The polynomial f earlier is not in the kernel of the map. Let's look at an example of a polynomial that does get mapped to zero.

Example 7.4

Let f be the polynomial $x_1x_3 - x_2^2$. When you apply the map $\widehat{\pi_A}$, we get $t_1(t_1t_2^2) - (t_1^2t_2^2) = 0$. So the polynomial f belongs to the kernel of the map $\widehat{\pi_A}$.

The point is that we should find all of the polynomials that are in the kernel. Here is the second observation: since I_A is an ideal,

Lemma 7.5

There exist finitely many polynomials $f_1,\ldots,f_s$ so that $I_A = \langle f_1,\ldots,f_s\rangle$.

The polynomials f_i will actually end up being binomials: you're not going to believe it!

The following is the main theorem.

Theorem 7.6

For the toric ideal I_A associated to the matrix A, the following are true:

(a) The ideal I_A is generated by finitely many binomials, i.e., by elements of the form

$$x_1^{u_1}x_2^{u_2}\cdots x_n^{u_n} - x_1^{v_1}x_2^{v_2}\cdots x_n^{v_n},$$

which we abbreviate by $x^u - x^v$, such that $Au = Av$.

(b) For every monomial order $\succ$, there exists a Gröbner basis G of the ideal I_A which is composed only of binomials.

Continuing the running example,

Example 7.7: R

member the four columns of A are labeled by $x_1,\ldots,x_4$ and the rows are indexed t_1 and t_2. Then, the ideal I_A is generated as follows:

$$I_A = \langle x_1x_3 - x_2^2, x_2x_4 - x_3^2, x_1x_4 - x_2x_3, x_1^2x_4 - x_2^3, x_1x_4^2 - x_3^3\rangle$$

The binomials belong in I_A, but they actually generate this ideal! We can show how to do this in `Macaulay2`.

For optimization, you might be thinking: how am I going to reconcile this with what I need to do to optimize? This is simple: you should think of this as a vector. For example, the square-free polynomial in the previous example should be associated with the vector $(1,-1,-1,1)$. Notice that this vector is in the integer kernel of A. Similarly, the next binomial in the list is associated to the vector $(2,-3,0,1)$. We should think of these as oriented vectors.

The beautiful thing here is that these are integer kernels. In linear algebra, you compute real kernels. Before we prove the next theorem, we need the following lemma:

Lemma 7.8

The toric ideal I_A associated to the matrix A is a K-vector space generated by all binomials of the form $x^u - x^v$ such that $Au = Av$. In other words, every polynomial $f \in I_A$ can be written as a linear combination of the form

$$\sum c_{uv}(x^u - x^v), \tag{7.1}$$

where $c_{uv} \in K$ and $x^u - x^v \in I_A$.

In our example, the rank of the matrix A is two, but this is the dimension of the real kernel, in the sense of linear algebra. But, I'm not talking about that space. I need to make a basis for the integer points in the null space, which makes it much more difficult. What I'm saying is that, if you give me a vector in the integral kernel of the matrix A, I will write it as the integer linear combination of the five vectors associated to the binomials above.

Let's prove Lemma **??**.

Proof. Note that $x^u - x^v \in I_A$ if and only if $Au = Av$. When you apply the map $\widehat{\pi_A}$, this is the same as doing $t^{Au} - t^{Av}$, which is 0. This follows from the definition of $\widehat{(}\pi_A)$.

We will prove this by contradiction. Suppose there is a polynomial $f \in I_A$ so that it cannot be written in the from (**??**). Using a monomial order $\succ$, assume that f is minimal in the sense that $LM_\succ(f)$ is the smallest possible. (We know this exists by the well-ordering property.) Our goal will be to find polynomial that is even "smaller" than f.

Now, we know that $f \in I_A$, which means that $\widehat{\pi_A}(f) = 0$. But that means that $f(t^{a_1}, t^{a_2}, \ldots, t^{a_n}) = 0$.

Let x^u be the leading monomial of f with respect to $\succ$. So, $\widehat{\pi_A}(x^u) = t^{Au}$. The only way to become zero is: There exists x^v term inside f that cancels with x^u. In other words, $t^{Av} = t^{Au}$. Remember that x^u was the leading term. But $x^u = LM_\succ(f) \succ x^v$. Define f' to be the polynomial $f -$ coefficient x^u in $f(x^u - x^v)$.

By the earlier observation, the binomial $x^u - x^v$ belongs to I_A, by the observation $t^{Av} = t^{Au}$. By the property of ideals, f' is in I_A. Here is our contradiction, f' has a smaller leading term than f does. ♠

If you want, you can actually obtain your contradiction in several ways. Example: since you can write f' in the form (**??**), then you get a contradiction for f. Therefore, f' cannot be written in the form of (**??**).

Now, we are ready to prove the main theorem. Proving part (a) of the theorem is super easy. You guys are probably going to laugh.

Proof. [Proof of the theorem] To prove part (a), by Hilbert's Basis Theorem, there exist polynomials $f_1, \ldots, f_s$ in I_A such that $I_A = \langle f_1, \ldots, f_s \rangle$. Recall here, that the coefficients are polynomials. But, by Lemma **??**, each polynomial f_i is of the form

$$f_i = \sum_{Au=Av} c_{uv}(x^u - x^v).$$

Moreover, each sum above is finite. Take for each i the binomials $x^u - x^v$ in the above expressions. That's the proof of part (a). The proof of (b) is not that much harder. The proof requires Buchberger's Algorithm.

How do you compute a Gröbner basis? By part (a), we know that the toric ideal I_A can be written as

$$I_A = \langle x^u - x^v \rangle,$$

where the generation is over finitely-many pairs of vectors satisfying $Au = Av$ (or, that is, $A(u - v) = 0$). How did Buchberger's Alogrithm go? We needed to compute S-pairs. Let's remind ourselves about S-pairs. What happens when you take the S-pair of two binomials $x^u - x^v$ and $x^\alpha - x^\beta$?

Let's assume that both binomials here are written in order, so that the leading monomials are first. Recall $S_\succ x^u - x^v, x^\alpha - x^\beta) = \frac{\text{lcm}(x^u, x^\alpha)}{x^u}(x^u - x^v) - \frac{\text{lcm}(x^u, x^\alpha)}{x^\alpha}(x^\alpha - x^\beta)$. But, we note that the inner terms here are

going to cancel. That is, we get $\frac{\text{lcm}(x^u,x^\alpha)}{x^u}(x^\nu) + \frac{\text{lcm}(x^u,x^\alpha)}{x^\alpha}(x^\beta)$. In particular, this expression is a binomial. (Note that this does not happen for general S-pairs. The punchline/moral of the story is that: S-pairs of binomials are themselves binomials.)

Now, when you do the Buchberger Algorithm, you need to divide the binomial by other binomials. When you divide binomials by binomials, you either get zero or another binomial. Therefore, when you apply the Buchberger Algorithm to the original set of binomials, you get a Gröbner basis that consists only of binomials. Therefore, there is a Gröbner basis that consists only of binomials. ♠

The moral of the story is that the vectors of $\ker(A) \cap Z^n$ precisely correspond to binomials in the toric ideal I_A. Thus, the Gröbner basis G of I_A is enough to express *any* vector in $\ker(A) \cap \mathbb{Z}^n$ as an integer linear combination of the elements of G.

Example 7.9

Consider the set of all 2×3 *matrices whose entries are all non-negative integers with the property that each row adds to* 6 *and each column adds to* 4*. An example of such a matrix is*

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}.$$

Another example is

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix}.$$

Finally, another is

$$\begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 4 \end{bmatrix}.$$

How many of them are there? There are 19. If you want to know which matrix is "cheapest" with respect to a cost vector, we're going to find out how to do that.

With toric ideals I_A, we will answer the following three fundamental questions for applications:

1. How many integer solutions are there for $\{x \mid Ax = b, x \in \mathbb{Z}^n, x \geq 0\}$?
2. Important for statistics: Can you find one such solution uniformly at random?
3. The question relevant to this class: Given a cost vector c, I wish to compute $x = \bar{x}$ in the set above that minimizes $c \cdot \bar{x}$.

Those questions depend on the previous theorem, but they also depend on the following definition and theorem.

Definition 7.10

Given a set of vectors $\mathscr{F} \subseteq \ker(A) \cap \mathbb{Z}^n$ and given the b-fiber $P_A(b) = \{\bar{x} \mid A\bar{x} = b, x \geq 0, x \in \mathbb{Z}^n\}$, I can define a graph $G_{\mathscr{F}}(P_A(b))$ as follows:

- *The vertices of the graph are exactly the elements of the fiber $P_A(b)$.*
- *Two elements u, v in the b-fiber $P_A(b)$ are connected by an edge exactly when $u - v \in \mathscr{F}$ or $v - u \in \mathscr{F}$.*

Let's look at an example of this graph. Let's do the example of the 2×3 matrices from before:

Example 7.11: T

e set $\mathscr{F}$ should consist of matrices that are in the kernel of the 5×6 matrix A of 2×3 transportation polytopes.
Recall that we had feasible points

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 4 \end{bmatrix}.$$

Consider a vector such that the row and column sums are zero. Using a vector like this, we get from the first matrix above to the second matrix above. Here is an example:

$$\begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix}$$

By adding this matrix to the the feasible solution on the left above, we obtain

$$\begin{bmatrix} 3 & 3 & 0 \\ 1 & 1 & 4 \end{bmatrix}.$$

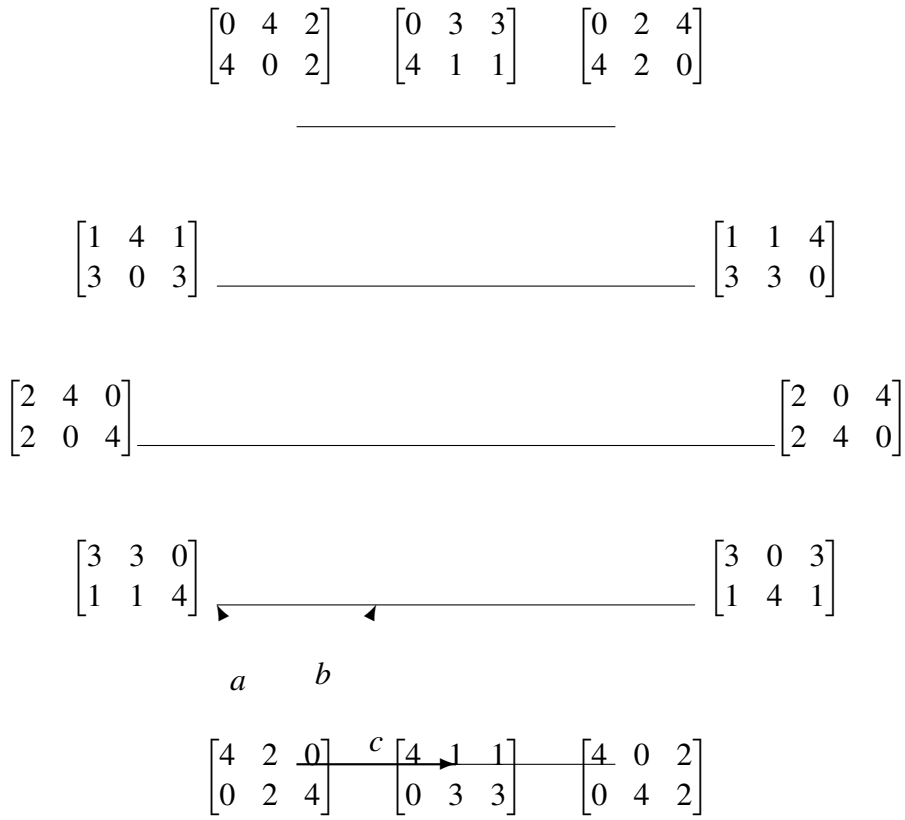
There are more steps to get from the matrix on the left to the matrix on the right. The goal is to have a set $\mathscr{F}$ of vectors that allows us to move between all feasible solutions via addition and subtraction.

For this problem, there are 3 basic moves. The moves are:

$$\pm\begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix}, \pm\begin{bmatrix} -1 & 0 & 1 \\ 1 & 0 & -1 \end{bmatrix} \text{ and } \pm\begin{bmatrix} 0 & -1 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

Let the positive versions of each of these be a, b, and c.

As mentioned, there are 19 tables, and here is a picture of the graph:


a
b
c

A. Linear Algebra

A.1 Contributors

lyryx
advancing learning [1]

Champions of Access to Knowledge

OPEN TEXT

All digital forms of access to our high-quality open texts are entirely FREE! All content is reviewed for excellence and is wholly adaptable; custom editions are produced by Lyryx for those adopting Lyryx assessment. Access to the original source files is also open to anyone!

ONLINE ASSESSMENT

We have been developing superior online formative assessment for more than 15 years. Our questions are continuously adapted with the content and reviewed for quality and sound pedagogy. To enhance learning, students receive immediate personalized feedback. Student grade reports and performance statistics are also provided.

SUPPORT

Access to our in-house support team is available 7 days/week to provide prompt resolution to both student and instructor inquiries. In addition, we work one-on-one with instructors to provide a comprehensive system, customized for their course. This can include adapting the text, managing multiple sections, and more!

INSTRUCTOR SUPPLEMENTS

Additional instructor resources are also freely accessible. Product dependent, these supplements include: full sets of adaptable slides and lecture notes, solutions manuals, and multiple choice question banks with an exam building tool.

[1] This book was not produced by Lyryx, but this book has made substantial use of their open source material. We leave this page in here as a tribute to Lyryx for sharing their content.

lyryx
advancing Learning

A First Course in Linear Algebra

an Open Text

BE A CHAMPION OF OER!

Contribute suggestions for improvements, new content, or errata:

A new topic

A new example

An interesting new question

A new or better proof to an existing theorem

Any other suggestions to improve the material

Contact Lyryx at info@lyryx.com with your ideas.

CONTRIBUTIONS

Ilijas Farah, York University

Ken Kuttler, Brigham Young University

Lyryx Learning Team

Foundations of Applied Mathematics

https://github.com/Foundations-of-Applied-Mathematics

CONTRIBUTIONS

List of Contributors

E. Evans
Brigham Young University

R. Evans
Brigham Young University

J. Grout
Drake University

J. Humpherys
Brigham Young University

T. Jarvis
Brigham Young University

J. Whitehead
Brigham Young University

J. Adams
Brigham Young University

J. Bejarano
Brigham Young University

Z. Boyd
Brigham Young University

M. Brown
Brigham Young University

A. Carr
Brigham Young University

C. Carter
Brigham Young University

T. Christensen
Brigham Young University

M. Cook
Brigham Young University

R. Dorff
Brigham Young University

B. Ehlert
Brigham Young University

M. Fabiano
Brigham Young University

K. Finlinson
Brigham Young University

J. Fisher
Brigham Young University

R. Flores
Brigham Young University

R. Fowers
Brigham Young University

A. Frandsen
Brigham Young University

R. Fuhriman
Brigham Young University

S. Giddens
Brigham Young University

C. Gigena
Brigham Young University

M. Graham
Brigham Young University

F. Glines
Brigham Young University

C. Glover
Brigham Young University

M. Goodwin
Brigham Young University

R. Grout
Brigham Young University

D. Grundvig
Brigham Young University

E. Hannesson
Brigham Young University

J. Hendricks
Brigham Young University

A. Henriksen
Brigham Young University

I. Henriksen
Brigham Young University

C. Hettinger
Brigham Young University

S. Horst
Brigham Young University
K. Jacobson
Brigham Young University
J. Leete
Brigham Young University
J. Lytle
Brigham Young University
R. McMurray
Brigham Young University
S. McQuarrie
Brigham Young University
D. Miller
Brigham Young University
J. Morrise
Brigham Young University
M. Morrise
Brigham Young University
A. Morrow
Brigham Young University
R. Murray
Brigham Young University
J. Nelson
Brigham Young University
E. Parkinson
Brigham Young University
M. Probst
Brigham Young University
M. Proudfoot
Brigham Young University
D. Reber
Brigham Young University
H. Ringer
Brigham Young University
C. Robertson
Brigham Young University
M. Russell
Brigham Young University
R. Sandberg
Brigham Young University
C. Sawyer
Brigham Young University
M. Stauffer
Brigham Young University
J. Stewart
Brigham Young University
S. Suggs
Brigham Young University
A. Tate
Brigham Young University
T. Thompson
Brigham Young University
M. Victors
Brigham Young University
J. Webb
Brigham Young University
R. Webb
Brigham Young University
J. West
Brigham Young University
A. Zaitzeff
Brigham Young University

This project is funded in part by the National Science Foundation, grant no. TUES Phase II DUE-1323785.

A.1.1. Graph Theory

Chapter on Graph Theory adapted from: CC-BY-SA 3.0 Math in Society A survey of mathematics for the liberal arts major Math in Society is a free, open textbook. This book is a survey of contemporary mathematical topics, most non-algebraic, appropriate for a college-level quantitative literacy topics course for liberal arts majors. The text is designed so that most chapters are independent, allowing the instructor to choose a selection of topics to be covered. Emphasis is placed on the applicability of the mathematics. Core material for each topic is covered in the main text, with additional depth available through exploration exercises appropriate for in-class, group, or individual investigation. This book is appropriate for Washington State Community Colleges' Math 107.
The current version is 2.5, released Dec 2017. http://www.opentextbookstore.com/mathinsociety/2.5/GraphTheory.pdf
Communicated by Tricia Muldoon Brown, Ph.D. Associate Professor of Mathematics Georgia Southern University Armstrong Campus Savannah, GA 31419http://math.armstrong.edu/faculty/brown/MATH1001.html