Applications of Linear Programing

András London

University of Szeged, Faculty of Informatics and Natural Sciences
Institute of Informatics, Department of Computational Optimization

2019

University of Szeged, Hungary
Venue: H-6720 Szeged, Dugonics square 13.
www.u-szeged.hu
www.szechenyi2020.hu

This teaching material has been made at the University of Szeged, and supported by the European Union. Project identity number: EFOP-3.4.3-16-2016-00014

Author: András London, PhD
Lecturers: András Pluhár, PhD, Zoltán Kincses, PhD
ISBN: 978-963-306-676-8
University of Szeged
2019, Szeged, Hungary

University of Szeged, Hungary
Venue: H-6720 Szeged, Dugonics square 13.
www.u-szeged.hu
www.szechenyi2020.hu

Applications of Linear Programming

András London

Preface

These lecture notes were written based on the lectures of the course *Application of Linear Programming* at the University of Szeged, and primarily dedicated as an excipient to that course and may serve as a textbook to similar courses. The *Application of Linear Programming* is an MSc course, the participants have already had introductory courses to Discrete Mathematics, Graph Theory, Operations Research, Probability Theory, etc. We do not assume deep knowledge on these subjects. In some places we do not give a detailed description/reminder, but expect some prerequisite knowledge at some places.

The subject of Linear Programming has its roots in the study of linear inequalities, which can be traced as far back to the work of Fourier in the early 19th century; a significant development of the field has started at the time of World War II. The United States Army launched a research group to support military operations using mathematical tools and models and the field Operations Research developed. A notable member of the group was George Dantzig who re-created the linear programming model and the developed the simplex algorithm to solve it. After WW II, when the first results were published, the number of applications quickly emerged. Even more, new mathematical fields emerged or re-boosted, such as Game Theory, Network Flow Theory, Nonlinear Programming, Stochastic Programming, Combinatorial Optimization, Semidefinite Optimization, etc.

In this notes we concentrate mainly on the models of some important real-life problems and some solution methods, and pay less attention to the classical (and deep) theoretical results and algorithms.

After providing a brief overview of the basic elements of Linear Programming in Chapters 1-3, we give an introduction to Integer Programming and present the Branch-and-Bound method by example in Chapter 4. In Chapters 5 and 7 we establish a connection between network flows and linear programming. The cornerstone is the notion and properties of totally unimodular matrices, discussed in Chapter 6, which enables us to get integer solutions in some cases. We also give an

introduction to Stochastic problems (Chapter 8), Game theory (Chapter 9) and their relation to mathematical programming. The efficiency of the described algorithms is also briefly discussed (Chapter 10). Finally we present the basics syntax of the AMPL language to help the Reader for solving problems by computer.

The notes were meant to provide a succinct summary of the material, most of which was loosely based on the lecture notes *Juraj Stacho: Introduction to Operations Research (Columbia University)*, the book *Wayne L. Winston: Operations Research*, the book *Robert R. Vanderbei: Linear Programming: Foundations and Extensions (Princeton University)* and the lecture notes *András Pluhár: Operations Research (University of Szeged)*. Some parts were inspired by other lecture notes (see cited Literature) and sources available on the Internet.

Finally, I would like to thank András Pluhár for carefully reading and lecturing this work. The lecture notes benefited greatly from his thoughtful comments and suggestions.

Contents

Chapter 1

Introduction

In this chapter we give a brief overview of the history, motivation and some important applications of Linear Programming. We introduce LP modeling by various examples.

1.1 Motivation: Why LP?

1.1.1 Brief history

The subject of linear programming has its roots in the study of linear inequalities, which can be traced back to the work of Fourier. The applied side of the subject got its start in 1939 when L.V. Kantorovich noted the practical importance of a certain class of linear programming problems and gave an algorithm for their solution. During World War II, he developed methods to plan expenditures and returns in order to reduce costs of the army and to increase losses imposed on the enemy. For several years, Kantorovich's work had been unknown in the West and unnoticed in the East. About the same time as Kantorovich, the Dutch-American economist T.C. Koopmans formulated classical economic problems as linear programs. Kantorovich and Koopmans later shared the 1975 Nobel Memorial Prize in Economics. In 1941, F.L. Hitchcock also formulated transportation problems as linear programs and gave a solution very similar to the simplex method invented in 1947 by George Dantzig for solving the linear programming problems that arose in U.S. Air Force planning problems. The earliest published accounts of Dantzig's work appeared in 1951. In the same year that Dantzig invented the simplex method, Koopmans showed that linear programming provided the appropriate model for the analysis of classical economic theories. Dantzig discussed his simplex method with János Neumann who immediately conjectured the

theory of duality by realizing that the problem he had been working in game theory was equivalent to what we call now Strong Duality Theorem. The linear programming problem was first shown to be solvable in polynomial time by Leonid Khachiyan in 1979, but a larger theoretical and practical breakthrough in the field came in 1984 when Karmarkar introduced a new interior point method for solving linear programming problems.

The field is still actively researched; currently applying of LP models effectively in various domains from logistic to energy market pricing, and developing efficient implementations of LP, IP (integer programming) and MIP (mixed integer programming) solvers are the biggest challenges in the field.

1.1.2 Applications

Linear programming is a widely used field of optimization for several reasons. Many practical problems in operations research can be expressed as linear programming problems. Certain special cases of linear programming, such as network flow problems are considered important enough to have generated much research on specialized algorithms for their solution. Linear programming was heavily used in the early formation of microeconomics and it is currently utilized in company management, such as planning, production, transportation, technology and other issues. Although the modern management issues are ever-changing, most companies would like to maximize profits and minimize costs with limited resources. It is widely known that many issues can be characterized as linear programming problem and the number of successful applications has been increasing continuously.

1.2 Modeling by example

1.2.1 Product mix

Example. A toy company makes two types of toys: toy soldiers and trains. Each toy is produced in two stages, first it is constructed in a carpentry shop, and then it is sent to a finishing shop, where it is varnished, vaxed, and polished.

To make one toy soldier costs $10 for raw materials and $14 for labor; it takes 1 hour in the carpentry shop, and 2 hours for finishing. To make one train costs $9 for raw materials and $10 for labor; it takes 1 hour in the carpentry shop, and 1 hour for finishing.

There are 80 hours available each week in the carpentry shop, and 100 hours for finishing. Each toy soldier is sold for \$27 while each train for \$21. Due to decreased demand for toy soldiers, the company plans to make and sell at most 40 toy soldiers; the number of trains is not restricted in any way.

What is the optimum (best) product mix (i.e., what quantities of which products to make) that maximizes the profit (assuming all toys produced will be sold)?

1.2.2 Notations

We try to code the given information into mathematical language. This is a crucial step in any application, so it is worth to spend time on it now and in the later examples too.

- **Decision variables**: $x_1, x_2, \ldots, x_i, \ldots$
- **Variable domains**: $x_1, x_1 \geq 0, \ldots, x_i \in \mathbb{R}, \ldots$
- **Goal/objective**: maximize/minimize
- **Objective function**: function to minimize/maximize: $2x_1 + 5x_2$
- **Constraints** (equations, inequalities): $3x_1 + 2x_2 \leq 10$

1.2.3 LP model for the product mix

Decision variables:

- x_1: number of soldiers will be produced
- x_2: number of trains will be produced

Goal: maximizing the profit

- $\$27 - \$10 - \$14 = \$3$ is the profit after selling a soldier $\Rightarrow 3x_1$ after selling x_1 soldiers
- $\$21 - \$9 - \$10 = \$2$: is the profit after selling a train $\Rightarrow 2x_2$ after selling x_2 trains

Objective function:

- $z = 3x_1 + 2x_2$: profit if selling x_1 soldiers and x_2 trains

Constraints:

- to produce x_1 soldiers and x_2 trains
  - $1x_1 + 1x_2$ hours are needed in the carpentry shops; there are at most 80 hours available
  - $2x_1 + 1x_2$ hours are needed for finishing; there are at most 100 hours available
- the number of soldiers, x_1, cannot be more than 40

Sign of variables x_1 and x_2 are non-negative (*and integer*).

Thus, we have formulated the following problem.

$$\begin{aligned} \max \quad z &= 3x_1 + 2x_2 \\ x_1 \quad &+ x_2 \leq 80 \\ 2x_1 \quad &+ x_2 \leq 100 \\ x_1 \quad &\leq 40 \\ x_1, x_2 &\geq 0 \end{aligned}$$

We call this system **program**. It is **linear**, since

- the objective function is a linear function of the decision variables,
- the constraints are linear inequalities (or equalities).

1.2.4 Formulating a linear program

Often, the following steps are used to formulate a linear program.

1. Choose **decision variables**
2. Choose an objective and an **objective function** that is a linear function of the decision variables
3. Choose **constraints** that are linear inequalities
4. Choose **sign restrictions** of variables

1.2.5 Basic definitions

The Linear Program (LP) in **standard form** (maximization) is

$$\begin{array}{lrcrcrcrcl} \max & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & = & z \\ \text{s.t.} & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \leq & b_2 \\ & \vdots & & & & & & & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \leq & b_m \\ & & & & & & & x_1, \dots, x_n & \geq & 0 \end{array}$$

A **feasible solution** (or feasible point) is a $p = (p_1, \dots, p_n) \in \mathbb{R}^n$ vector, such that putting p_i to x_i $(\forall i = 1, \dots, n)$ p satisfies the all constraints of the LP. The set of feasible solutions, also called **feasible region** is the set of all feasible points.

The **optimal solution** is a feasible solution with the maximum value of the objective function.

1.3 Solving linear programs

1.3.1 Graphical method

We may plot (in 2D or 3D) each constraint as an equation (that is a line in the plane in 2D). In case of '$\leq$' constraints, feasible points are on one side of the line, see Fig. 4.3.

A **corner (extreme) point** X of the region R: every line through X intersects R in a segment whose one endpoint is X. Solving a linear program amounts to finding a best corner point by the following theorem.

Theorem 1.1 *If a linear program has an optimal solution, then it also has an **optimal solution** that is a **corner point** of the feasible region.*

Exercise. Try to find all corner points of the feasible region of the product mix problem. Evaluate the objective function $3x_1 + 2x_2$ at those points.

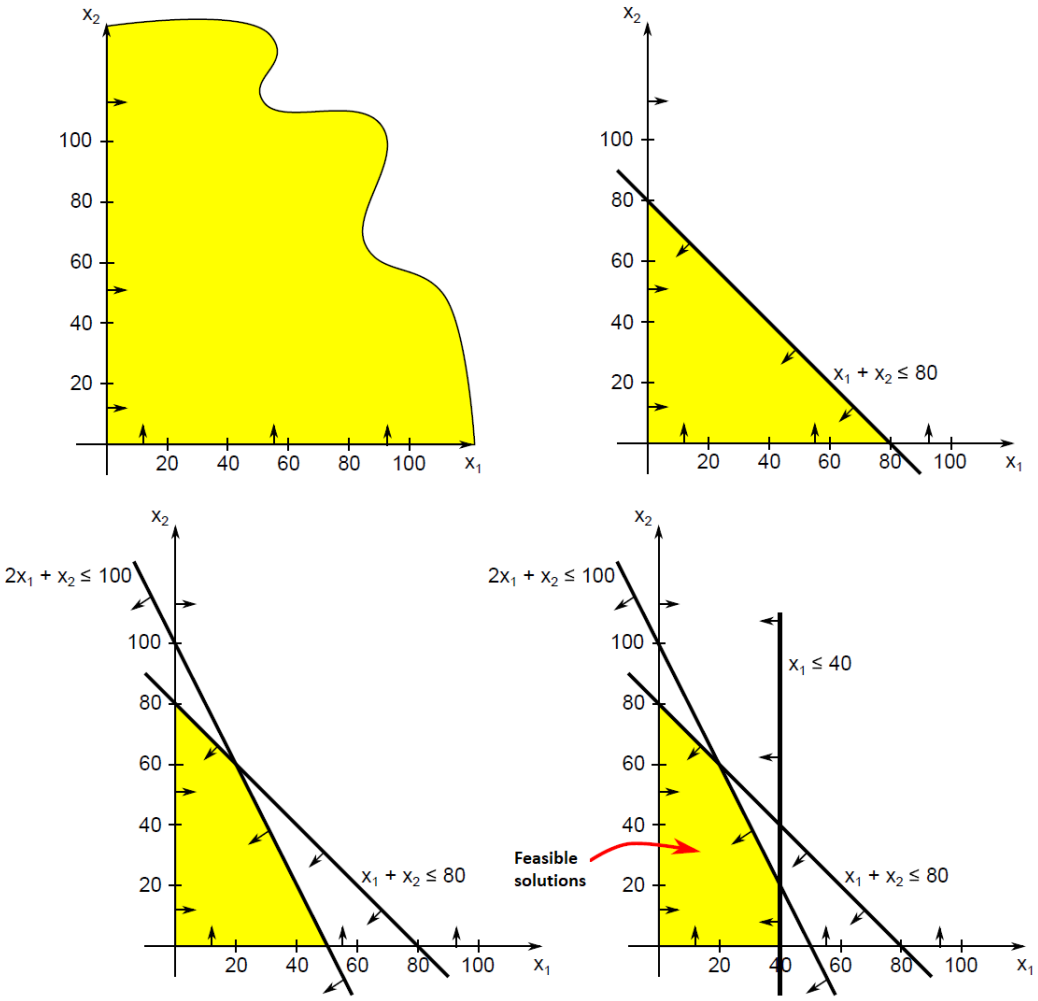


Figure 1.1: Set of feasible solutions of the product mix problem. Source: Juraj Stacho's lecture notes

Note that the graphical solving process always yields one of the following cases.

Theorem 1.2 *Every linear program has either*

1. a **unique** optimal solution, or
2. multiple (**infinity**) optimal solutions, or
3. is **infeasible** (i.e. has no feasible solution), or
4. is **unbounded** (i.e. no feasible solution is maximal).

1.4 More examples

1.4.1 Blending

A company wants to produce a certain alloy containing 30% lead, 30% zinc, and 40% tin. This is to be done by mixing certain amounts of existing alloys that can be purchased at certain prices. The company wishes to minimize the cost. There are nine available alloys with the following composition and prices.

Alloy	1	2	3	4	5	6	7	8	9	Blend
Lead (%)	20	50	30	30	30	60	40	10	10	30
Zinc (%)	30	40	20	40	30	30	50	30	10	30
Tin (%)	50	10	50	30	40	10	10	60	80	40
Cost ($ / kg)	7.3	6.9	7.3	7.5	7.6	6.0	5.8	4.3	4.1	minimize

Decision variables are $x_1, x_2, \ldots, x_9$, where x_i is the amount of Alloy i in a unit of blend. In particular, the decision variables must satisfy $x_1 + x_2 + \cdots + x_9 = 1$. (It is a common mistake to choose x_i the absolute amount of Alloy i in the blend. That may lead to a non-linear program.) With that we can setup constraints and the objective function.

$$
\begin{aligned}
\min \quad z = {} & 7.3x_1 + 6.9x_2 + 7.3x_3 + 7.5x_4 + 7.6x_5 + 6.0x_6 + 5.8x_7 + 4.3x_8 + 4.1x_9 \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 = 1 \\
& 0.2x_1 + 0.5x_2 + 0.3x_3 + 0.3x_4 + 0.3x_5 + 0.6x_6 + 0.4x_7 + 0.1x_8 + 0.1x_9 = 0.3 \\
& 0.3x_1 + 0.4x_2 + 0.2x_3 + 0.4x_4 + 0.3x_5 + 0.3x_6 + 0.5x_7 + 0.3x_8 + 0.1x_9 = 0.3 \\
& 0.5x_1 + 0.1x_2 + 0.5x_3 + 0.3x_4 + 0.4x_5 + 0.1x_6 + 0.1x_7 + 0.6x_8 + 0.8x_9 = 0.4 \\
& x_1, \ldots, x_9 \geq 0
\end{aligned}
$$

Do we need all the four equations?

1.4.2 Post office problem

In a post office (or in any vegetable shop) the following number of workers are needed from Monday to Sunday, respectively): M:27 T:24 W:23 T:20 F:25 Sa:27 Su:28 (this is necessary to run the business, but we are flexible, means that a bit more employees in each day is not a big problem). Further requirement that an employee must work in 5 consecutive days, e.g. from Monday

to Friday, from Tuesday to Saturday, etc. The task of the manager to satisfy all conditions with a minimum number of employees (therefore minimize the wage cost).

Let the decision variables be $x_1, \ldots, x_7$, where x_i is the number of employees start working on day i ($i = 1, \ldots, 7$ for Monday $i = 1$, etc.).

With that we can setup constraints and the objective function.

$$
\begin{aligned}
\min \quad z = {} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\
& x_1 + x_4 + x_5 + x_6 + x_7 \geq 27 \\
& x_1 + x_2 + x_5 + x_6 + x_7 \geq 24 \\
& x_1 + x_2 + x_3 + x_6 + x_7 \geq 23 \\
& x_1 + x_2 + x_3 + x_4 + x_7 \geq 20 \\
& x_1 + x_2 + x_3 + x_4 + x_5 \geq 25 \\
& x_2 + x_3 + x_4 + x_5 + x_6 \geq 27 \\
& x_3 + x_4 + x_5 + x_6 + x_7 \geq 28 \\
& x_i \geq 0 (i = 1, \ldots, 7)
\end{aligned}
$$

1.5 Exercises

1.5.1 (Winston, Ch.3. Problems) Leary Chemical manufactures three chemicals: A, B, and C. These chemicals are produced via two production processes: 1 and 2. Running process 1 for an hour costs \$4 and yields 3 units of A, 1 of B, and 1 of C. Running process 2 for an hour costs \$1 and produces 1 unit of A and 1 of B. To meet customer demands, at least 10 units of A, 5 of B, and 3 of C must be produced daily. Graphically determine a daily production plan that minimizes the cost of meeting Leary Chemical's daily demands.

1.5.2 (Winston, Ch.3. Problems) Furnco manufactures desks and chairs. Each desk uses 4 units of wood, and each chair uses 3. A desk contributes \$40 to profit, and a chair contributes \$25. Marketing restrictions require that the number of chairs produced be at least twice the number of desks produced. If 20 units of wood are available, formulate an LP to maximize Furnco's profit. Then graphically solve the LP.

1.5.3 Furniture company manufactures four models of chairs. Each chair requires certain amount of raw materials (wood/steel) to make. The company wants to decide on a production that maximizes profit (assuming all produced chair are sold). The required and available amounts of mate-

rials are as follows.

	Chair 1	Chair 2	Chair 3	Chair 4	Total available
Steal	1	1	3	9	4400 (kg)
Wood	4	9	7	2	600 (kg)
Profit	\$12	\$20	\$18	\$40	max

1.5.4 The Bloomington Brewery Inc. produces pilsner type beers and ale type beers. The price of the pilsner type is 40 euros per barrel, while the price of the ale type is 30 euros per barrel. To produce one barrel pilsner they need 5 kg barely malt and 2 kg hop. On the other hand, to produce one barrel ale 3 kg barely malt and 1 kg hop are needed. In a given day, they have 60 kg barley malt and 25 kg hop in storage. The task is to determine the production numbers in order to maximize the expected income.

1.5.5 The Dorian Cars company distribute luxury cars and lorries/trucks. The management observed that the costumers are mostly men and women with high income. To reach this group the company is going to start an advertising campaign and buy 30 seconds advertising space during two kind of TV broadcasts: comedies and football games. Estimations show that each commercial during a comedy is watched by 7 million women (with high salary) and 2 million men, while in case of a football game the numbers are 2 millions and 12 millions, respectively. The cost of a commercial is 5,000 dollars in case of a comedy and 10,000 dollars in case of a football game. The ownership wants that the commercials would be seen by at least 28 million women and 24 million men reached by the minimum commercial cost.

Chapter 2

Simplex Method

In this chapter we present the simplex method as it applies to linear programming problems in standard form. At the end of the chapter we briefly discuss the connection of LP with convex geometry.

2.1 Simplex algorithm

Let us consider again the following LP:

$$\begin{array}{lrcrcrcl} \text{max} & z & = & 3x_1 & + & 2x_2 & & \\ \text{S.t.} & & & x_1 & + & x_2 & \leq & 80 \\ & & & 2x_1 & + & x_2 & \leq & 100 \\ & & & x_1 & & & \leq & 40 \\ & & & & & x_1, x_2 & \geq & 0 \end{array}$$

To change an inequalities to an equations, we add a new **non-negative variable** to each constraint. We call these **slack variables** or **artificial variables**.

$$\begin{array}{lrcrcrcrcrcl} \text{max} & z & = & 3x_1 & + & 2x_2 & & & & & \\ \text{S.t.} & & & x_1 & + & x_2 & + & x_3 & & & = & 80 \\ & & & 2x_1 & + & x_2 & & & + & x_4 & = & 100 \\ & & & x_1 & & & & & & + \ x_5 & = & 40 \\ & & & x_1, & & x_2, & & x_3, & & x_4, \quad x_5 & \geq & 0 \end{array}$$

Now we can express the slack variables from the individual equations and obtain

$$\begin{array}{rcrcrcr} x_3 & = & 80 & - & x_1 & - & x_2 \\ x_4 & = & 100 & - & 2x_1 & - & x_2 \\ x_5 & = & 40 & - & x_1 & & \\ \hline z & = & 0 & + & 3x_1 & + & 2x_2 \end{array}$$

This is called the first **dictionary**. In the general case, the LP is

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & + & x_{n+1} & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & + & x_{n+2} & = & b_2 \\ & & & & \vdots & & & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \ldots & + & a_{mn}x_n & + & x_{n+m} & = & b_m \\ \hline c_1x_1 & + & c_2x_2 & + & \ldots & + & c_nx_n & & & = & z \end{array}$$

and the corresponding first dictionary is defined as

$$\begin{array}{ccccccccccc} x_{n+1} & = & b_1 & - & a_{11}x_1 & - & a_{12}x_2 & - & \ldots & - & a_{1n}x_n \\ x_{n+2} & = & b_2 & - & a_{21}x_1 & - & a_{22}x_2 & - & \ldots & - & a_{2n}x_n \\ & & & & & \vdots & & & & & \\ x_{n+m} & = & b_m & - & a_{m1}x_1 & - & a_{m2}x_2 & - & \ldots & - & a_{mn}x_n \\ \hline z & = & & & c_1x_1 & + & c_2x_2 & + & \ldots & + & c_nx_n \end{array}$$

We will use the following definitions and terminology:

- **Decision variables**: variables of the original LP (given in standard form) $(x_1, x_2, \ldots, x_n)$.
- **Slack variables**: new non-negative variables to construct the dictionary $(x_{n+1}, x_{n+2}, \ldots, x_{n+m})$.
- **Basic variables**: variables on the left-hand side of the constraint equalities of the dictionary. The set of basic variables is called **basis**.
- **Non-basic variables**: variables on the right-hand side of the constraint equalities of the dictionary.
- **Basic solution**: vector x, such that the non-basic-variables are zero (and the basic variables are constant values of the respective equations on the right-hand side).

- **Feasible basic solution**: a basic solution which is feasible, i.e. $b_i \geq 0$ $i = 1, 2, \ldots, m$ is satisfied.

The importance of basic solutions is revealed by the observation that they are precisely the corner points of the feasible region. We have discussed that to find an optimal solution to an LP, it suffices to find a best solution among all corner points. The above tells us how to compute them since they are the basic feasible solutions.

At first we suppose that $b_1 \geq 0, \ldots, b_m \geq 0$, thus the basic solution of the initial dictionary is $x = (0, 0, \ldots, 0, b_1, b_2, \ldots, b_m)$ is a feasible solution. The algorithm we are seeking is an **iterative search of the optimal solution** that in each iteration tries to find a new dictionary such that

1. Every two consecutive dictionaries are equivalent

2. In each iteration the value of the objective function is larger or equal to that in case the previous dictionary

3. The basic solution is feasible in each iteration

The following terminology is used:

- **Pivot step**: calculate a new feasible by changing the roles of a basic and non-basic variable (i.e. re-arranging a constraint equation).

- **Incoming variable**: a non-basic variable that become a basic variable of the new dictionary in a simplex iteration.

- **Outgoing variable**: a basic variable that become a non-basic variable of the new dictionary in a simplex iteration.

Two **dictionaries are equivalent** if the two have the same set of solutions with equal objective function values for the same solutions.

Proposition 2.1 *Each dictionary, obtained by using the above defined pivot steps, is equivalent to the original system.*

2.1.1 Improving the solution

In our initial dictionary, x_1 and x_2 are non-basic variables, x_3, x_4 and x_5 are basic variables and the set $\{x_3, x_4, x_5\}$ is the basis.

$$\begin{array}{rcrcrcr} x_3 & = & 80 & - & x_1 & - & x_2 \\ x_4 & = & 100 & - & 2x_1 & - & x_2 \\ x_5 & = & 40 & - & x_1 & & \\ \hline z & = & 0 & + & 3x_1 & + & 2x_2 \end{array}$$

The basic (feasible) solution of this dictionary is $x_1 = 0$, $x_2 = 0$ and then $x_3 = 80, x_4 = 100, x_5 = 40$, and $z = 0$. How can we improve the solution, i.e. the value of z? We can see, that if x_1 or x_2 increases then z increases.

- For example consider $x_1 = 20$ and $x_2 = 0$. Then $x_3 = 60, x_4 = 60, x_5 = 20$, and $z = 60$. This is a feasible solution.
- Now let $x_1 = 40$ and $x_2 = 0$. Then $x_3 = 40, x_4 = 20, x_5 = 0$, and $z = 120$. This is also a feasible solution and better then the previous one.
- Next we consider $x_1 = 50$ and $x_2 = 0$. Then $x_3 = 30, x_4 = 0, x_5 = -10$, but this is not a feasible solution since $x_5 < 0$.

How much we can increase x_1 before a (dependent) basic variable becomes negative?
Let $x_1 = t$ and $x_2 = 0$. The the basic solution is feasible if

$$\begin{array}{rcrcrcrcl} x_3 & = & 80 & - & t & - & x_2 & \geq 0 & \Rightarrow \; t \leq 80 \\ x_4 & = & 100 & - & 2t & - & x_2 & \geq 0 & \Rightarrow \; t \leq 50 \\ x_5 & = & 40 & - & t & & & \geq 0 & \Rightarrow \; t \leq 40 \end{array}$$

The maximal value is $x_1 = 40$ at which point the variable x_5 becomes zero. Thus we perform a pivot step by changing the role of x_1 (incoming variable) and x_5 (outgoing variable). The above analysis can be easily done using the following simple **"ratio" test**.

- the ratio for x_3 is $80/1 = 80$
- the ratio for x_4 is $100/2 = 50$
- the ratio for x_5 is $40/1 = 40$

The minimum ratio achieved with x_5, thus x_5 will be the outgoing variable. After the first pivot step, we arrive the the following dictionary:

$$\begin{array}{rcrcrcr} x_1 & = & 40 & & & - & x_5 \\ x_3 & = & 40 & - & x_2 & + & x_5 \\ x_4 & = & 20 & - & x_2 & + & 2x_5 \\ \hline z & = & 120 & + & 2x_2 & - & 3x_5 \end{array}$$

The basic solution of this dictionary is $x_2 = 0, x_5 = 0$; $x_1 = 40, x_3 = 40, x_4 = 20$ and $z = 120$. Now we can observe that z can be increased by increasing x_2. The ratio test tells us that x_4 will be the outgoing variable (notice that the first equation does not limit the value of x_4). After the second pivot step we get

$$\begin{array}{rcrcrcr} x_1 & = & 40 & & & - & x_5 \\ x_2 & = & 20 & - & x_4 & + & 2x_5 \\ x_3 & = & 20 & + & x_4 & - & x_5 \\ \hline z & = & 160 & - & 2x_4 & + & x_5 \end{array}$$

with the basic solution s $x_4 = 0, x_5 = 0$; $x_1 = 40, x_2 = 20, x_3 = 20$ and $z = 160$. Now we can increase x_5. According to the ratio test x_3 is the outgoing variable, and we get the dictionary

$$\begin{array}{rcrcrcr} x_1 & = & 20 & + & x_3 & - & x_4 \\ x_2 & = & 60 & - & 2x_3 & + & x_4 \\ x_5 & = & 20 & - & x_3 & + & x_4 \\ \hline z & = & 180 & - & x_3 & - & x_4 \end{array}$$

We can see that no more improvement is possible (since increasing x_3 or x_4 would decrease z). The basic solution of this dictionary is the optimal solution of the LP: $x_1 = 20, x_2 = 60$ ($x_3 = 0, x_4 = 0, x_5 = 20$) and $z = 180$.

In the general case, the following proposition can be proved.

Proposition 2.2 *If there is no positive c_j ($j = 1, 2, \ldots, n+m$) coefficient in the objective function and negative b_i ($i = 1, 2, \ldots, m$) constant term in the equations, then the basic solution of the dictionary is optimal.*

2.1.2 Unbounded LP

Suppose that after an iteration step we arrived to the following dictionary.

$$\begin{array}{rcrcrcrcr} x_4 & = & 4 & + & x_1 & - & 2x_2 & & \\ x_5 & = & 6 & + & x_1 & & & - & 4x_3 \\ x_6 & = & 2 & & & - & 2x_2 & + & 2x_3 \\ \hline z & = & & & 3x_1 & - & 4x_2 & - & x_3 \end{array}$$

We can observe, that x_1 can be arbitrarily large in the actual basic solution, and so z can be arbitrarily large.

A maximization (minimization) LP is **unbounded** if its objective function value can be arbitrarily large over the set of feasible solutions. In general the following statement holds.

Proposition 2.3 *If there is a positive c_j ($j = 1, 2, \ldots, n + m$) coefficient in the equation for z such that all $-a_{ij}$ ($i = 1, 2, \ldots, m$) coefficient of x_j in the equations is non-negative, then the corresponding LP is unbounded.*

2.1.3 Simplex algorithm

Preparation: find a starting feasible solution/dictionary

1. Convert LP to canonical form (constraints are equalities) by adding slack variables $x_{n+1}, \ldots, x_{n+m}$
2. Construct a starting dictionary - express slack variables and objective function z
3. If the resulting dictionary is feasible, then we are done with preparation
   If not, try to find a feasible dictionary using the Phase I. method (see later)

Simplex step (maximization LP): try to improve the solution

1. Optimality test: If **no variable** appears with a **positive** coefficient in the equation for $z \rightarrow$ STOP, current solution is **optimal**
   - set non-basic variables to zero
   - read off the values of the basic variables (these are the constant terms in respective equations) and the objective function z
   - report this optimal solution

2. Else pick a variable x_i having positive coefficient in the equation for z; x_i is the **incoming variable**

3. Ratio test: in the dictionary, find an equation for a variable x_j in which

   - x_i appears with a negative coefficient $-a$
   - the ratio b/a is the smallest possible (b is the constant term in the equation for x_j)

4. If no such such x_j exists $\rightarrow$ stop, no optimal solution, report that **LP is unbounded**

5. Else x_j is the **outgoing variable** $\rightarrow$ construct a new dictionary by pivoting:

   - express x_i from the equation for x_j
   - add this as a new equation
   - remove the equation for x_j
   - substitute x_i to all other equations (including the one for z)

6. Repeat from 1.

2.2 Cycling, degeneracy

Now we will see, that it can happen that in optimality test of the simplex algorithm the answer is always no, means that the algorithm does not stop in any iteration. Let us run the simplex algorithm on the following dictionary.

- First iteration

$$\begin{array}{rcrrrrrrrrrr}
\boxed{x_5} &=& & - & \frac{1}{2}x_1 & + & \frac{11}{2}x_2 & + & \frac{5}{2}x_3 & - & 9x_4 \\
x_6 &=& & - & \frac{1}{2}x_1 & + & \frac{3}{2}x_2 & + & \frac{1}{2}x_3 & - & x_4 \\
x_7 &=& 1 & - & x_1 & & & & & & \\
\hline
z &=& & & \boxed{10x_1} & - & 57x_2 & - & 8x_3 & - & 24x_4
\end{array}$$

- Second iteration

$$\begin{array}{rcrrrrrrrrrr}
x_1 &=& & & 11x_2 & + & 5x_3 & - & 18x_4 & - & 2x_5 \\
\boxed{x_6} &=& & - & 4x_2 & - & 2x_3 & + & 8x_4 & + & x_5 \\
x_7 &=& 1 & - & 11x_2 & - & 5x_3 & + & 18x_4 & + & x_5 \\
\hline
z &=& & & \boxed{53x_2} & + & 41x_3 & - & 204x_3 & - & 20x_5
\end{array}$$

- Third iteration

$$\begin{array}{rcrcrcrcrcr}
\boxed{x_1} & = & & - & \frac{1}{2}x_3 & + & 4x_4 & + & \frac{3}{4}x_5 & - & \frac{11}{4}x_6 \\
x_2 & = & & - & \frac{1}{2}x_3 & + & 2x_4 & + & \frac{1}{4}x_5 & - & \frac{1}{4}x_6 \\
x_7 & = & 1 & + & \frac{1}{2}x_3 & - & 4x_4 & - & \frac{3}{4}x_5 & - & \frac{11}{4}x_6 \\
\hline
z & = & & & \boxed{\frac{29}{2}x_3} & - & 98x_4 & - & \frac{27}{4}x_5 & - & \frac{53}{4}x_6
\end{array}$$

- (...)

- 7th iteration

$$\begin{array}{rcrcrcrcrcr}
\boxed{x_5} & = & & - & \frac{1}{2}x_1 & + & \frac{11}{2}x_2 & + & \frac{5}{2}x_3 & - & 9x_4 \\
x_6 & = & & - & \frac{1}{2}x_1 & + & \frac{3}{2}x_2 & + & \frac{1}{2}x_3 & - & x_4 \\
x_7 & = & 1 & - & x_1 & & & & & & \\
\hline
z & = & & & \boxed{10x_1} & - & 57x_2 & - & 8x_3 & - & 24x_4
\end{array}$$

We get back our initial dictionary after 7 iterations. Several dictionaries may correspond to the same (degenerate) solution The simplex (pivot) rule may **cycle**, it is possible to go back to the same dictionary.

Proposition 2.4 *If the simplex algorithm fails to terminate, then it must cycle.*

Note that the number of possible dictionaries is $\binom{n+m}{m}$ finite. The simplex algorithm changes the dictionaries in each iteration, thus if it does not stop, then it must arrive to a dictionary in a certain step that appeared before. It is crucial to see that the following proposition holds

Proposition 2.5 *If the basis of two dictionaries are the same, then the two dictionaries are identical.*

We say that a dictionary is **degenerate** if b_j vanishes for some $j = 1, \ldots, m$. A degenerate dictionary could cause difficulties for the simplex algorithm. The problem is that the simplex method could make a sequence of degenerate pivots (i.e. choosing an outgoing variable x_j where $b_j = 0$) and eventually return to a dictionary that has appeared before.

Proposition 2.6 *Cycling caused by degenerate pivots.*

Fortunately, there are strategies to avoid cycling. For instance, **Bland's rule** says that from possible options, choose an incoming (outgoing) variable x_k with smallest index k.

Theorem 2.1 *The simplex method always terminates provided that both the entering and the leaving variable are chosen according to Bland's rule.*

Another option is e.g. the *lexicographic simplex method*: choose as outgoing variable one whose row is lexicographically smallest (when divided by the constant term) - the coefficients in the objective function are guaranteed to strictly increase lexicographically.

2.3 Two-phase simplex method

We defined the dictionary for **LP in standard (maximization) form** as

$$\begin{aligned} x_{n+i} &= b_i - \sum_{j=1}^{n} a_{ij} x_j \qquad\qquad i = 1, 2, \ldots, m \\ \hline z &= \sum_{j=1}^{n} c_j x_j \end{aligned}$$

We assumed till now that every $b_i \geq 0$ $i = 1, 2, \ldots, m$ and solved the problem with the simplex algorithm. What happens, if it is not the case? For instance, let us consider the following LP:

$$\begin{array}{lrcrcrcl} \max & z = & x_1 & - & x_2 & + & x_3 & \\ & & 2x_1 & - & x_2 & + & x_3 & \leq 4 \\ & & 2x_1 & - & 3x_2 & + & x_3 & \leq -5 \\ & & -x_1 & + & x_2 & - & 2x_3 & \leq -1 \\ & & & & & x_1, & x_2, x_3 & \geq 0 \end{array}$$

The corresponding dictionary is

$$\begin{array}{rcrcrcrcr} x_4 &=& 4 &-& 2x_1 &+& x_2 &-& 2x_3 \\ x_5 &=& -5 &-& 2x_1 &+& 3x_2 &-& x_3 \\ x_6 &=& -1 &+& x_1 &-& x_2 &+& 2x_3 \\ \hline z &=& & & x_1 &-& x_2 &+& x_3 \end{array}$$

This is **not feasible** since $x_5, x_6 < 0$ in the basic solution. We need to find a starting feasible dictionary. In order to do this, we solve a different problem. **Idea**: introduce one new artificial

variable x_0 and a new objective $w = -x_0$, and solve the following **auxiliary problem** .

$$\begin{array}{lrcrcrcrcl} \max & w = & & & & & & -x_0 & & \\ & & 2x_1 & - & x_2 & + & x_3 & - & x_0 & \leq 4 \\ & & 2x_1 & - & 3x_2 & + & x_3 & - & x_0 & \leq -5 \\ & & -x_1 & + & x_2 & - & 2x_3 & - & x_0 & \leq -1 \\ & & & & & & & x_1, x_2, & x_3, x_0 & \geq 0 \end{array}$$

Introduce x_4, x_5, x_6 slack variables as before, we get

$$\begin{array}{lrcrcrcrcrcl} \max & w = & & & & & & -x_0 & & & & \\ & 2x_1 & - & x_2 & + & x_3 & - & x_0 & + & x_4 & = 4 & \\ & 2x_1 & - & 3x_2 & + & x_3 & - & x_0 & + & x_5 & = -5 & \\ & -x_1 & + & x_2 & - & 2x_3 & - & x_0 & + & x_6 & = -1 & \\ & & & & & & & & & x_i & \geq 0 & i = 1, \ldots 6 \end{array}$$

Now perform the following steps:

1. consider the inequality whose right-hand side is most negative (in this case 2nd inequality)
2. this inequality has an associated slack variable (x_5), remove this variable from basic variables $\rightarrow \{x_4, x_6\}$
3. add x_0 in place of the removed variable $\rightarrow \{x_0, x_4, x_6\}$

Lemma 2.1 *The auxiliary problem become feasible after the above (simple rearranging) steps.*

We arrive to the dictionary

$$\begin{array}{rcrcrcrcrcr} x_0 & = & 5 & + & 2x_1 & - & 3x_2 & + & x_3 & + & x_5 \\ x_4 & = & 9 & & & - & 2x_2 & & & + & x_5 \\ x_6 & = & 4 & + & 3x_1 & - & 4x_2 & + & 3x_3 & + & x_5 \\ \hline w & = & -5 & - & 2x_1 & + & 3x_2 & - & x_3 & - & x_5 \end{array}$$

and this is **feasible**! (for the auxiliary problem).

Lemma 2.2 *The standard LP has a feasible solution if and only if $w = 0$ is the optimal solution of the auxiliary problem.*

Proof. Suppose that x is a feasible solution of the original problem. Then $(x_0 = 0, x)$ is an optimal solution of the auxiliary problem and $w\left((x_0 = 0, x)\right) = 0$. Conversely, suppose that 0 is the optimum of the auxiliary problem taken in x^*. Then $x_0^* = 0$, and leaving x_0^* from x^* we obtain a feasible solution of the original problem □

According to Lemma 2.2 the steps of Phase I of the simplex method can be summarized as follows.

1. If the basic solution of the corresponding dictionary of the LP in is feasible, then go to Phase II. (simplex algorithm).
2. If not, associate the auxiliary problem and prepare its initial feasible dictionary.
3. Solve the auxiliary problem with the simplex algorithm.
4. If its optimum < 0, then there is no feasible solution of the original problem.
5. If its optimum $= 0$, then we get a dictionary that is equivalent with the dictionary of the original LP and its basic solution is feasible. Then
   - drop the variable $x_0(= 0)$ and remove the auxiliary objective w;
   - introduce the original objective z;
   - if there are variables in z that are basic variables, substitute them using the new dictionary.

2.4 Fundamental theorem of linear programming

Now that we have a Phase I algorithm and a variant of the simplex method that is guaranteed to terminate, we can summarize the main result of this chapter.

Theorem 2.2 (Fundamental theorem of linear programming) *For an arbitrary linear program in standard form, the following statements are true:*

1. *If there is no optimal solution, then the problem is either infeasible or unbounded.*
2. *If a feasible solution exists, then a basic feasible solution exists.*

3. *If an optimal solution exists, then a basic optimal solution exists.*

Proof. The Phase I algorithm either proves that the problem is infeasible or results in a basic feasible solution. The Phase II algorithm either discovers that the problem is unbounded or finds a basic optimal solution. These statements depend, of course, on applying a variant of the simplex method that does not cycle, which exists as we noted before. □

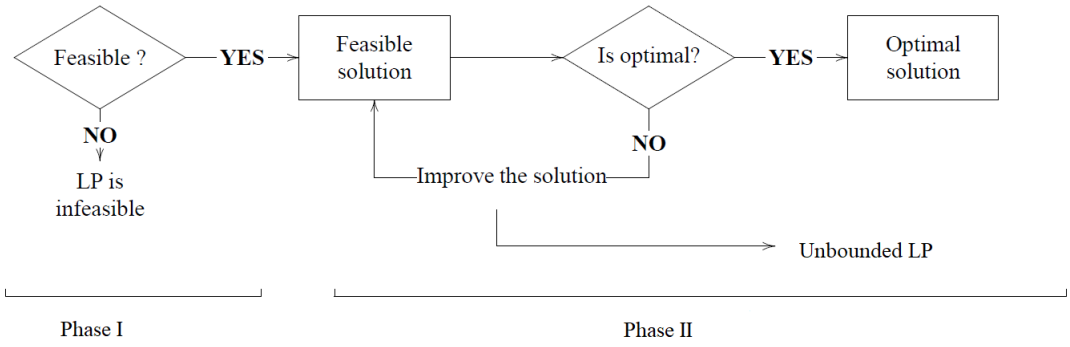


Figure 2.1: Two-phase simplex method. Source: Juraj Stacho's lecture notes

2.5 Connection to convex geometry

Let $\mathbb{R}^n$ be the n-dimensional **linear space** over the real numbers. Its elements are real **vectors** of n elements. Let $\mathbb{E}^n$ be the n-dimensional **Euclidean space**, with an **inner product** operation and a **distance** function that are defined as follows

- inner product: $\langle x, y \rangle = x^T y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$
- corresponding norm: $||x|| = \sqrt{\langle x, x \rangle}$
- distance: $d(x, y) = ||x - y||_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$

This distance function is called the *Euclidean metric*. The formula expresses a special case of the *Pythagorean theorem*.

A **Point** is an $x \in \mathbb{E}^n$ vector. Note that **LP feasible solutions** are points in $\mathbb{E}^n$. The n-**dimensional hyperplane** is defined as

$$\{\, x \;:\; x \in \mathbb{E}^n,\; a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = b \,\},$$

where $a_1, a_2, \ldots, a_n, b \in \mathbb{R}$ given (fixed) numbers.

The n**-dimensional closed half-space** is defined as

$$\{\, x \;:\; x \in \mathbb{E}^n,\; a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b \,\},$$

where $a_1, a_2, \ldots, a_n, b \in \mathbb{R}$ given (fixed) numbers.

The **linear constraints** of an LP are closed half-spaces (in case of '$\leq$' constraints) or hyperplanes (in case of '$=$' constraints). The **feasible region** (i.e. the set of feasible solutions) of a linear program is the intersection of finite number of half-spaces (and hyperplanes). This is called **polyhedron**.

A **polytope** is a bounded polyhedron. The set of feasible solutions (points) of a linear program forms a convex polyhedron (either bounded or unbounded). Theorem 1.1 tells us that a linear objective function achieves its maximal value (if exists) in a corner (extreme) point of the feasible region (i.e. a polytope).

Example. Let us consider the following LP:

$$\begin{array}{lrcrcrcl} \text{Max} & z & = & 2x_1 & + & x_2 & + & 5x_3 \\ \text{S.t.} & & & 2x_1 & + & x_2 & & \leq\ 4 \\ & & & & & & x_3 & \leq\ 3 \\ & & & x_1, & & x_2, & x_3 & \geq\ 0 \end{array}$$

Performing the simplex algorithm on this LP using the classical pivoting rule (the entering variable is the one with the largest positive coefficient) we obtain the following basic solutions:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 4 & 5 \end{bmatrix} \to \begin{bmatrix} 0 & 0 & 5 & 4 & 0 \end{bmatrix} \to \begin{bmatrix} 2 & 0 & 5 & 0 & 0 \end{bmatrix} \to \begin{bmatrix} 0 & 4 & 5 & 0 & 0 \end{bmatrix}.$$

Exercise. Verify that by solving the problem with the simplex algorithm.

The basic solutions for the decision variables are

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \to \begin{bmatrix} 0 & 0 & 5 \end{bmatrix} \to \begin{bmatrix} 2 & 0 & 5 \end{bmatrix} \to \begin{bmatrix} 0 & 4 & 5 \end{bmatrix}$$

that are the corner points of the polyhedron represents the feasible solutions, see Figure 2.2. A pivot step is a step from an actual basic solution (corner point) to an "adjacent" basic solution (corner point) through an egde of a polyhedron; this is guaranteed by the above theorems.

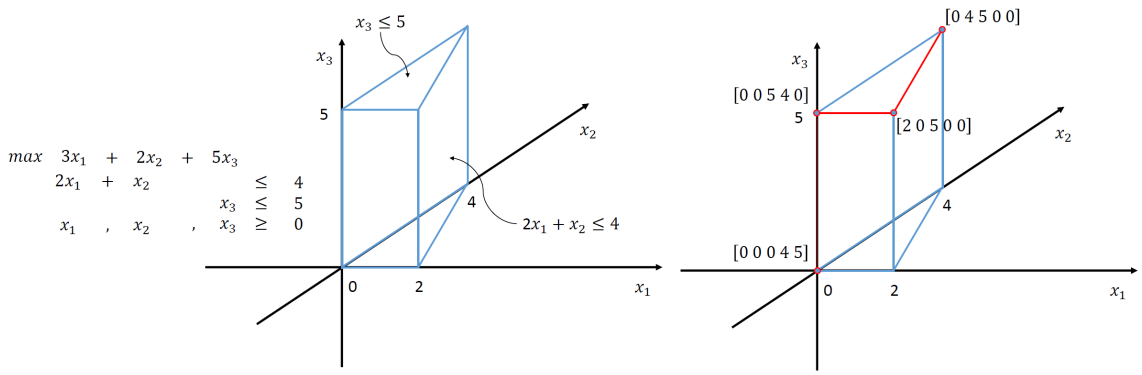


Figure 2.2: Set of feasible solutions (polyhedron), basic solutions (corner points) and pivot steps (red)

Degeneracy can also be explained in terms of geometry. It means that an n-dimensional corner point is incident to at least $n + 1$ hyperplanes (it is easy to consider in the plain, when a vertex is the intersection of at least 3 lines). Degenerate iteration step means, that the algorithm remains in the same vertex. The basis chances, but the basic solution remains the same. We learnt that using an appropriate pivot rule, cycling caused by degeneracy can be avoided.

Also note, that if an LP is **unbounded** then the set of feasible solutions (i.e. a polyhedron) is unbounded, but the reverse direction of this statement is not follows.

2.6 Exercises

2.6.1 Solve the following linear programming problem:

$$\begin{array}{lrcrcrcl} \max & z & = & 2x_1 & + & x_2 & & \\ \text{S.t.} & & & 2x_1 & + & x_2 & \leq & 4 \\ & & & 2x_1 & + & 3x_2 & \leq & 3 \\ & & & 4x_1 & + & x_2 & \leq & 5 \\ & & & x_1 & + & 5x_2 & \leq & 1 \\ & & & & & x_1, x_2 & \geq & 0 \end{array}$$

2.6.2 Solve the following linear programming problem:

$$
\begin{array}{lrcrcrcl}
\max & z & = & x_1 & + & 3x_2 & & \\
\text{S.t.} & & & -x_1 & - & 3x_2 & \leq & -3 \\
& & & -x_1 & + & x_2 & \leq & -1 \\
& & & x_1 & + & 2x_2 & \leq & 4 \\
& & & & & x_1, x_2 & \geq & 0
\end{array}
$$

2.6.3 Solve the following linear programming problem:

$$
\begin{array}{lrcrcrcrcll}
\max & z & = & 6x_1 & + & 8x_2 & + & 5x_3 & + & 9x_4 & & \\
\text{S.t.} & & & x_1 & + & x_2 & + & x_3 & + & x_4 & = & 1 \\
& & & & & & & & & x_i & \geq & 0 \quad (i = 1, 2, 3, 4)
\end{array}
$$

2.6.4 Give an example showing that the variable that becomes basic in one iteration of the simplex method can become nonbasic in the next iteration.

2.6.5 Show that the variable that becomes nonbasic in one iteration of the simplex method cannot become basic in the next iteration.

2.6.6 Prove the propositions and theorems of the chapter.

Chapter 3

Duality

In this chapter we give a short introduction to duality theory. With each linear program another problem can be associated, called its dual. The dual of this dual linear program is the original linear program. Hence, linear programs come in primal/dual pairs. It turns out that the objective function of every feasible solution for one of these two linear programs gives a bound on the value of optimal objective function for the other. Moreover, if one of them has an optimal solution then the its pair has too, and the optimal objective function values are equal.

3.1 Duality: pricing interpretation

Let us go back to our usual manufacturing LP problem. For the sake of illustration, we drop the 3rd constraint, and consider the resource items as blocks of wood and cans of paint (instead of working hours).

$$\begin{aligned} \max \quad z = 3x_1 + 2x_2 & & \text{[profit]} \\ x_1 + x_2 & \leq 80 & \text{[wood]} \\ 2x_1 + x_2 & \leq 100 & \text{[paint]} \\ x_1, x_2 & \geq 0 & \end{aligned}$$

The manufacturer owns 80 blocks of wood and 100 cans of paint. *He can sell his stock at market prices or buy additional stock at market prices*. He can also produce and sell goods (toys) using the available stock. **What is his best strategy** (assuming everything produced can be sold)?

Let y_1 be the (market) price of one block of wood and y_2 be the price of one can of paint. The manufacturer can

1. Produce toys using the available resources and sell them.

2. Sell his resources in market prices.

3. Buy additional wood and paint in market prices.

Selling stock generates a profit of $80y_1 + 100y_2$. If the cost (in market prices) of producing one toy soldier is strictly less than the sale price, i.e. if $y_1 + 2y_2 < 3$, then there is *no limit on the profit* of manufacturer. He can generate arbitrarily large profit by buying additional stock to produce toy soldiers in arbitrary amounts. *Why*?

The production cost of one toy soldier is $y_1 + 2y_2$, thus producing x_1 toy soldier costs $(y_1 + 2y_2)x_1$. Suppose that $y_1+2y_2 = 2.9\$$, and the selling price is $3\$$. Therefore the profit by selling one toy soldier is $0.1\$$ and the profit by selling x_1 is $0.1x_1$, that can be arbitrarily large. The situation is similar in the case of trains.

In long term, **market** (the market competition) will "not allow" the manufacturer to make arbitrarily large profit (*why and how?*). It will set its prices so that the manufacturer makes as little as possible. The market thus "solves" the following:

$$\begin{array}{lrcrcl} \min & 80y_1 & + & 100y_2 & & \\ & y_1 & + & 2y_2 & \geq 3 & \text{[soldiers]} \\ & y_1 & + & y_2 & \geq 2 & \text{[trains]} \\ & & & y_1, y_2 & \geq 0 & \end{array}$$

The above LP is called the **dual** of the manufacturing problem. In general, the **primal-dual pair** is defined as

$$\begin{array}{lrl} & \sum_{j=1}^{n} a_{ij}x_j \leq b_i & i = 1, 2, \ldots m \\ \textbf{Primal} & x_j \geq 0 & j = 1, 2, \ldots n \\ \hline & \max \sum_{i=1}^{n} c_i x_i = z & \end{array}$$

$$\sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad j = 1, 2, \ldots n$$

Dual $\qquad y_i \geq 0 \qquad i = 1, 2, \ldots m$

$$\min \sum_{i=1}^{m} b_i y_i = w$$

Writing them in matrix form the primal is

$$\begin{array}{rrcl} \max & c^T x & = & z \\ & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

while its dual pair is

$$\begin{array}{rrcl} \min & b^T y & = & w \\ & A^T y & \geq & c \\ & y & \geq & 0 \end{array}$$

Observe that the dual can be obtained easily form the primal (in standard form):

- transposing matrix A
- swapping vectors b and c
- switching the $\leq$ inequalities to $\geq$
- changing $\max$ to $\min$.

Proposition 3.1 *The dual of the dual is the primal LP.*

Proof. Re-writing the dual to standard maximization form we get

$$\sum_{i=1}^{m} (-a_{ij}) y_i \leq -c_j \qquad j = 1, 2, \ldots n$$

Dual $\qquad y_i \geq 0 \qquad i = 1, 2, \ldots m$

$$\max \sum_{i=1}^{m} (-b_i) y_i = w$$

$$\sum_{j=1}^{n}(-a_{ij})x_j \geq -b_i \qquad i = 1, 2, \ldots m$$

Dual of dual $\qquad x_j \geq 0 \qquad j = 1, 2, \ldots n$

$$\min \sum_{j=1}^{n} -(c_j)x_j = z$$

that is equivalent to the standard primal LP. □

Finally here, consider that our LP is the *model of a production process to maximize profit with limited resources* (like our toy example). In this problem m is the *number of resources*, n is the *number of different products*. The *number of units produced* from product j is x_j. Producing one unit of product j requires a_{ij} units of resource i, while b_i is the available quantity (units) of resource i. The profit of selling one unit of product j is c_j

In the dual optimal solution y_i^* is the **marginal price** (or also called **shadow price**) of resource i (in the primal). Increasing the quantity b_i with one unit the objective function value (of the primal) is increasing by y_i^*. On the other hand, if there is "too much" form resource i then the value of this resource cannot be high. Moreover one may observe that it does not worth to pay more than y_i^* to a unit of resource i.

3.2 Duality theorems and feasibility

Theorem 3.1 (Weak duality) *If* $x = (x_1, \ldots, x_n)$ *is* feasible solution *for the primal and* $y = (y_1, \ldots, y_m)$ *is* feasible solution *for the dual, then* $c^T x \leq b^T y$*, i.e.*

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i y_i.$$

It means that any feasible solution of the dual is an upper bound of all feasible solution (also the primal optimum, in particular) of the primal.

Proof. The proof is a simple chain of obvious inequalities:

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} y_i a_{ij} \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} x_j a_{ij} \right) y_i \leq \sum_{i=1}^{m} b_i y_i,$$

or in matrix form:

$$c^T x \leq (A^T y)^T x = (y^T A)x = y^T(Ax) \leq y^T b = b^T y.$$

□

Theorem 3.2 (Strong duality) *If the primal* has an optimal solution $x = (x_1, \ldots, x_n)$ *then the dual* has an optimal solution $y = (y_1, \ldots, y_m)$ *such that* $c^T x = b^T y$*, i.e.*

$$\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i.$$

Furthermore the following equalities hold:

$$y^T(b - Ax) = 0 \text{ and } x^T(A^T y - c) = 0.$$

Simply, if i*th* constraint inequality is not sharp *(no equality) in the primal optimum, then* the corresponding dual y_i variable is 0*. Backwards, if primal* x_i *variable is strictly positive in the optimum, then the corresponding dual constraint is sharp (equality holds). This is called* **complementary slackness**.

As a consequence of the duality theorems, if primal is unbounded, then dual must be infeasible and likewise, if dual is unbounded, then primal must be infeasible. Note that is it possible that both primal and dual are infeasible. But if both are feasible, then neither of them is unbounded.

Primal / Dual	No feasible solution	Feasible solution exists	Unbounded
No feasible solution	✓	×	✓
Feasible solution exists	×	✓	×
Unbounded	✓	×	×

Proof. We omit the details of the proof. The complementary slackness property can be seen easily as

$$0 \leq y^T(b - Ax) = y^T b - y^T Ax = b^T y - (A^T y)^T x \leq b^T y - c^T x = 0,$$

and

$$0 \leq x^T(A^T y - c) = (y^T A - c^T)x = y^T(Ax) - c^T x \leq y^T b - c^T x = b^T y - c^T x = 0.$$

□

3.3 Dual simplex method

The dual simplex method is simply a new way of choosing the entering and leaving variables in a sequence of primal dictionaries. Let us consider the following example:

$$\begin{array}{llrcrcrcr} \text{m} & z & = & -x_1 & - & x_2 & & \\ \text{s.t.} & & & -2x_1 & - & x_2 & \leq & 4 \\ & & & -2x_1 & + & 4x_2 & \leq & -8 \\ & & & -x_1 & + & 3x_2 & \leq & -7 \\ & & & & & x_1, x_2 & \geq & 0. \end{array}$$

The dual problem is

$$\begin{array}{llrcrcrcrcr} \text{min} & w & = & 4y_1 & - & 8y_2 & - & 7y_3 & & \\ \text{s.t.} & & & -2y_1 & - & 2y_2 & - & y_3 & \geq & -1 \\ & & & -y_1 & + & 4y_2 & + & 3y_3 & \geq & -1 \\ & & & & & & & y_1, \; y_2, y_3 & \geq & 0. \end{array}$$

Introducing slack variables $x_3, x_4, x_5 \geq 0$ for the primal, and $y_4, y_5 \geq 0$ for the dual and transforming the dual to standard maximization problem we obtain the following dictionaries

$$\begin{array}{rcrcrcr} x_3 & = & 4 & + & 2x_1 & + & x_2 \\ x_4 & = & -8 & + & 2x_1 & - & 4x_2 \\ x_5 & = & -7 & + & x_1 & - & 3x_2 \\ \hline z & = & 0 & - & x_1 & - & x_2 \end{array}$$

for the primal, and

$$\begin{array}{rcrcrcrcr} y_4 & = & 1 & - & 2y_1 & - & 2y_2 & - & y_3 \\ y_5 & = & 1 & - & y_1 & + & 4y_2 & + & 3y_3 \\ \hline -w & = & 0 & - & 4y_1 & + & 8y_2 & + & 7y_3 \end{array}$$

for the dual. Note that the dual dictionary is feasible, but the primal is not. This suggest that it could be useful to apply the simplex algorithm for the dual and apply the "corresponding"pivots to the primal using the complementary slackness property.

The entering variable in the dual is y_2 and the leaving variable is y_4. Since x_4 is complementary to y_2 and x_1 is complementary to y_4, we use x_4 and x_1 for the leaving and entering variables in the

primal. The results of the pivots are

$$\begin{array}{rcrcrcr} x_3 & = & 12 & + & x_4 & + & 5x_2 \\ x_1 & = & 4 & + & 0.5x_4 & + & 2x_2 \\ x_5 & = & -3 & + & 0.5x_4 & - & x_2 \\ \hline z & = & -4 & - & 0.5x_4 & - & 3x_2 \end{array}$$

and

$$\begin{array}{rcrcrcrcr} y_2 & = & 0.5 & - & y_1 & - & 0.5y_4 & - & 0.5y_3 \\ y_5 & = & 3 & - & 5y_1 & - & 2y_4 & + & y_3 \\ \hline -w & = & 4 & - & 12y_1 & - & 4y_4 & + & 3y_3. \end{array}$$

Going on with the work on the dual, now y_3 is the entering and y_2 is the leaving variable.Thus, in the primal we use x_5 and x_4 as the leaving and entering variables, respectively. We get

$$\begin{array}{rcrcrcr} x_3 & = & 18 & + & 2x_5 & + & 7x_2 \\ x_1 & = & 7 & + & x_5 & + & 3x_2 \\ x_4 & = & 6 & + & 2x_5 & + & 2x_2 \\ \hline z & = & -7 & - & x_5 & - & 4x_2 \end{array}$$

and

$$\begin{array}{rcrcrcrcr} y_3 & = & 1 & - & 2y_1 & - & y_4 & - & 2y_2 \\ y_5 & = & 4 & - & 7y_1 & - & 3y_4 & - & 2y_2 \\ \hline -w & = & 7 & - & 18y_1 & - & 7y_4 & - & 6y_y. \end{array}$$

We notice that both dictionaries are optimal. Observe that in each of the above dictionaries, the table of numbers in each dual dictionary is the negative-transpose of the corresponding primal table. Therefore, it is not needed to write the dual dictionary; the dual simplex method can be entirely de- scribed in terms of the primal dictionaries. The procedure is the following:

1. **Select the leaving variable** by choosing that basic variable whose constant term in the dictionary is the most negative.
   - If there are none, then the current dictionary is optimal.
2. **Ratio test**.
   - Select the entering variable by scanning across this row of the dictionary and comparing ratios of the coefficients in this row to the corresponding coefficients in the objective row, looking for the largest negative ratio (just as in the primal simplex algorithm)
3. **Pivot** to the next dictionary and go to 1.

3.4 General LP and its dual

LP contains equalities or unrestricted variables can be also handled. Concretely, *equality* constraint corresponds to an *unrestricted variable*, and vice-versa (in primal and its dual pair, respectively). Suppose that $x_1 + x_2 = 80$. Then

$$3x_1 + 2x_2 \leq 5x_1 + 2x_2 = (-1)\underbrace{(x_1 + x_2)}_{=80} + 3\underbrace{(2x_1 + x_2)}_{\leq 100} \leq -80 + 3 \times 100 = 220$$

thus now $y_1 = -1$, but is in unrestricted. On the other hand, suppose that x_1 has no sign restriction. Then we could not conclude, for instance, that $x_1 + 2x_2 \leq 4x_1 + 2x_2$ holds for all feasible solutions. In the dual we had $y_1 + 2y_2 \geq 3$, and if we fix $y_1 + 2y_2 = 3$, then $(y_1 + 2y_2)x_1 + 2x_2 \leq 3x_1 + tx_2$ (for all $t > 2$) upper bound always works. In a similar way we can conclude that $\geq$ constraint corresponds to a non-positive variable and vice-versa. In general the following rules can be summarized:

Primal (Max)	**Dual** (Min)
i-th constraint $\leq$	$y_i \geq 0$
i-th constraint $\geq$	$y_i \leq 0$
i-the constraint $=$	y_i unrestricted
$x_i \leq 0$	i-th constraint $\leq$
$x_i \geq 0$	i-th constraint $\geq$
x_i unrestricted	i-th constraint $=$

Example. The primal LP is

$$\begin{array}{lrcrcrcl} \max & z = & 3x_1 & + & 2x_2 & + & x_3 & \\ & & x_1 & + & x_2 & + & 0.5x_3 & \leq 80 \\ & & 2x_1 & + & x_2 & + & x_3 & = 100 \\ & & x_1 & & & + & x_3 & \geq 40 \\ & & & & & & x_1 & \text{unrestricted} \\ & & & & & & x_2 & \leq 0 \\ & & & & & & x_3 & \geq 0 \end{array}$$

then its dual is

$$\begin{array}{lrcrcrcl}
\min & w = & 80y_1 & + & 100y_2 & + & 40y_3 & \\
 & & y_1 & + & 2y_2 & + & y_3 & = 3 \\
 & & y_1 & + & y_2 & & & \leq 2 \\
 & & 0.5y_1 & + & y_2 & + & y_3 & \geq 1 \\
 & & & & & & y_1 & \geq 0 \\
 & & & & & & y_2 & \text{unrestricted} \\
 & & & & & & y_3 & \leq 0
\end{array}$$

3.5 Complementary slackness

Recall that strong duality theorem says that if x is an optimal solution to the primal and y is an optimal solution to the dual, then $c^T x = b^T y$. In fact, more is true.

Theorem 3.3 ***(Complementary slackness)*** *Assume that x is an optimal solution to the primal. Then*

1. *If y is an* optimal *solution to the dual, then x and y are* complementary.
2. *If y is a* feasible *solution in the dual and is* complementary *to x, then y is* optimal *in the dual.*
3. *There* exists a feasible *solution y to the dual such that x and y are* complementary.

A consequence is of this theorem is the following.

Proposition 3.2 *If x is a basic feasible primal solution and π is the vector contains the corresponding shadow prices, then x is optimal if and only if π is a feasible solution of the dual.*

To understand the relevance of Theorem 3.3 let us consider the following LP:

$$\begin{array}{lrcrcrcrcl}
\max & z = & 6x_1 & + & x_2 & - & x_3 & - & x_4 & \\
 & & x_1 & + & 2x_2 & + & x_3 & + & x_4 & \leq 5 \\
 & & 3x_1 & + & x_2 & - & x_3 & & & \leq 8 \\
 & & & & x_2 & + & x_3 & + & x_4 & = 1 \\
 & & & & & & & & x_1 & \text{unrestricted} \\
 & & & & & & x_2, & x_3, & x_4 & \geq 0
\end{array}$$

We would like to check if one of the following assignments is an optimal solution

1. $x_1 = 2, x_2 = 1, x_3 = 0, x_4 = 0$

2. $x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0$

To check this, we need the dual LP:

$$\begin{array}{lrrrrrrl}
\min & w = & 5y_1 & + & 8y_2 & + & y_3 & \\
 & & y_1 & + & 3y_2 & & & = 6 \\
 & & 2y_1 & + & y_2 & + & y_3 & \geq 1 \\
 & & y_1 & - & y_2 & + & y_3 & \geq -1 \\
 & & y_1 & & & + & y_3 & \geq -1 \\
 & & & & & y_1, & y_2 & \geq 0 \\
 & & & & & & y_3 & \text{unrestricted}
\end{array}$$

The **first suggestion** is: $x_1 = 2, x_2 = 1, x_3 = 0, x_4 = 0$; suppose that this is optimal. Then there is a $y = (y_1, y_2, y_3)$ feasible solution that is complementary to x.

- the first primal constraint: $x_1 + 2x_2 + x_3 + x_4 = 2 + 2 + 0 + 0 = 4 < 5$ **not tight** $\rightarrow y_1 = 0$
- the second primal constraint: $3x_1 + x_2 - x_3 = 6 + 1 - 0 = 7 < 8$ **not tight** $\rightarrow y_2 = 0$
- then the first dual constraint: $y_1 + 3y_2 = 0 + 0 = 0 \neq 6 \rightarrow (y_1, y_2, y_3)$ **not feasible** solution of the dual, but we assumed that it is $\Rightarrow x$ **not optimal** solution of the primal

The **second suggestion** is: $x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0$; suppose that this is optimal. hen there is a $y = (y_1, y_2, y_3)$ feasible solution that is complementary to x.

- the first primal constraint $x_1 + 2x_2 + x_3 + x_4 = 3 + 0 + 1 + 0 = 4 < 5$ **not tight** $\rightarrow y_1 = 0$
- the second primal constraint: $3x_1 + x_2 - x_3 = 9 + 0 - 1 = 8$ **tight**
- the third primal constraint: $x_2 + x_3 + x_4 = 0 + 1 + 0 = 1$ **tight**
- check sign constraints $(x_1, x_2, x_3, x_4 \geq 0) \Rightarrow x$ **feasible** in the primal

Now check the values in x with respect to the dual constraints

- x_1 is unbounded $\rightarrow$ the first dual constraint $y_1 + 3y_2 = 6$ **is tight** (by necessity)
- $x_3 > 0 \rightarrow$ the third dual constraint is **tight**: $y_1 - y_2 + y_3 = -1$

Summarizing the above we get:

$$\begin{aligned} y_1 & & & & & = 0 \\ y_1 & + & 3y_2 & & & = 6 \\ y_1 & - & y_2 & + & y_3 & = -1 \end{aligned}$$

The unique solution of this system is: $y_1 = 0, y_2 = 2, y_3 = 1$; this is complementary to $x = (x_1, x_2, x_3, x_4)$ by construction. The last step is to check if y is also **feasible** in the dual. The answer is true, thus x is **optimal solution** of the primal.

3.5.1 Complementary slackness: summary

1. Given x, check if x is feasible;
2. find which variables y_1 should be 0;
3. find which dual constraints should be tight;
4. this yields a system of equations;
5. solve the system;
6. verify that the solution is feasible in the dual.

If all these steps succeed, then the given x is optimal; otherwise, it is not.

3.6 Exercises

3.6.1 What is the dual of the following linear programming problem:

$$\begin{aligned} \text{max} \quad z &= 2x_1 + x_2 \\ \text{S.t.} \quad & 2x_1 + x_2 \leq 4 \\ & 2x_1 + 3x_2 \leq 3 \\ & 4x_1 + x_2 \leq 5 \\ & x_1 + 5x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

3.6.2 What is the dual of the following linear programming problem:

$$\begin{array}{lrcrcrcl} \text{max} & z & = & x_1 & + & 3x_2 & & \\ \text{S.t.} & & & -x_1 & - & 3x_2 & \leq & -3 \\ & & & -x_1 & + & x_2 & \leq & -1 \\ & & & x_1 & + & 2x_2 & \leq & 4 \\ & & & & & x_1, x_2 & \geq & 0 \end{array}$$

3.6.3 What is the dual of the following linear programming problem:

$$\begin{array}{lrcrcrcrcl} \text{Max} & z & = & 6x_1 & + & 8x_2 & + & 5x_3 & + & 9x_4 & & \\ \text{S.t.} & & & x_1 & + & x_2 & + & x_3 & + & x_4 & = & 1 \\ & & & & & & & & & x_i & \geq & 0 \quad (i = 1, 2, 3, 4) \end{array}$$

3.6.4 Find an available LP Solver a check the primal and dual solutions of the above tasks.

3.6.5 (Diet problem). A master student was trying to make ends meet on a very small stipend. He went to the library and looked up "Recommended Dietary Allowances" and was able to determine a minimum daily intake quantity of each essential nutrient for a male in his weight and age category. Let m denote the number of nutrients that he identified as important to his diet, and let b_i for $i = 1, \ldots, m$ denote his personal minimum daily requirements. Next, he made a list of his favorite foods (which, except for pizza and due mostly to laziness and ineptitude in the kitchen, consisted almost entirely of frozen prepared meals). He then went to the local grocery store and made a list of the unit price for each of his favorite foods. Let us denote these prices as c_j for $j = 1, \ldots, n$. In addition to prices, he also looked at the labels and collected information about how much of the critical nutrients are contained in one serving of each food. Let us denote by a_{ij} the amount of nutrient i contained in food j. (Fortunately, he was able to call his favorite pizza delivery service and get similar information from them.)

- In terms of this information, formulate a linear program minimizes the student's cost.
- Make your on diet. Consider at least 5 nutrients and 7 different foods.
- Formulate the dual of the problem.
- Can you introduce another person into the above story whose problem would naturally be to solve the dual?

Chapter 4

Integer Programming

A (mixed) integer programming problem (IP or MIP) is an LP in which (some) all of the variables are required to be non-negative integers. Many real-life situations may be formulated as IPs and MIPs, but unfortunately, we will in this chapter that IPs are usually much harder to solve than LPs. We begin with necessary definitions and then present the so-called Branch-and-Bound technique to solve IPs. We also discuss the basics of dynamic programming and some practical solutions that an be formulated as IPs.

4.1 Introduction to IP

An LP in which all variables are required to be integers is called a **pure integer programming** (IP) problem. For example,

$$\begin{array}{lrcrcl} \max z = & 3x_1 & + & 2x_2 & & \\ \text{st} & x_1 & + & x_2 & \leq & 6 \\ & & & x_1, x_2 & \geq & 0 \\ & & & x_1, x_2 & & \text{integer} \end{array}$$

An IP in which only some of the variables are required to be integers is called a **mixed integer programming problem** (MIP). In the example above let $x_1, x_2 \geq 0$ and x_2 be an integer (x_1 is not required to be an integer). An integer programming problem in which all the variables must equal 0 or 1 is called a $0-1$ **IP**.

The LP obtained by **omitting all integer or** $0-1$ **constraints** on variables is called the **LP relaxation** of the IP.

Proposition 4.1 *The following statements holds.*

- *The feasible region for any IP must be contained in the feasible region of the corresponding LP relaxation.*
- *Optimal objective function value for LP relaxation $\geq$ optimal value for IP (for max problems).*
- *If all corner points of the feasible region of the LP- relaxation is integer, then it has an integer optimal solution that is also optimal solution of the corresponding IP.*
- *The optimal solution of the LP-relaxation can be arbitrarily far from the IP solution.*

4.2 The Branch & Bound method

The first method to solve IPs is based on the the **divide and conquer principle**. A large problem is divided into a few smaller ones. (This is the "branch" part.) The conquering part is done by estimate how good a solution obtained for each smaller problems (to do this, we may have to divide the problem further, until we get a problem that we can handle); that is the "bound" part.

We will use the linear programming **relaxation** to estimate the optimal solution of an integer programming. The steps of the method are the following:

1. Divide a problem into sub-problems
2. Calculate the LP relaxation of a sub-problem
    - The LP problem has no feasible solution, done;
    - The LP problem has an integer optimal solution; done. Compare the optimal solution with the best solution we know (the **incumbent**).
    - The LP problem has an optimal solution that is worse than the incumbent, done.
3. In all the cases above, we know all we need to know about that sub-problem. We say that sub-problem is **fathomed**.
4. The LP problem has an optimal solution that are not all integer, better than the incumbent. In this case go to 1 and repeat.

Example. We are given the following IP[1]:

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 5 \\ & & & x_1, x_2 & \geq & 0 \\ & & & x_1, x_2 & & \text{integer} \end{array}$$

We illustrate how the B&B method works. The set of feasible solutions is shown if Fig. 4.1 The LP relaxation of the problem is

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 5 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

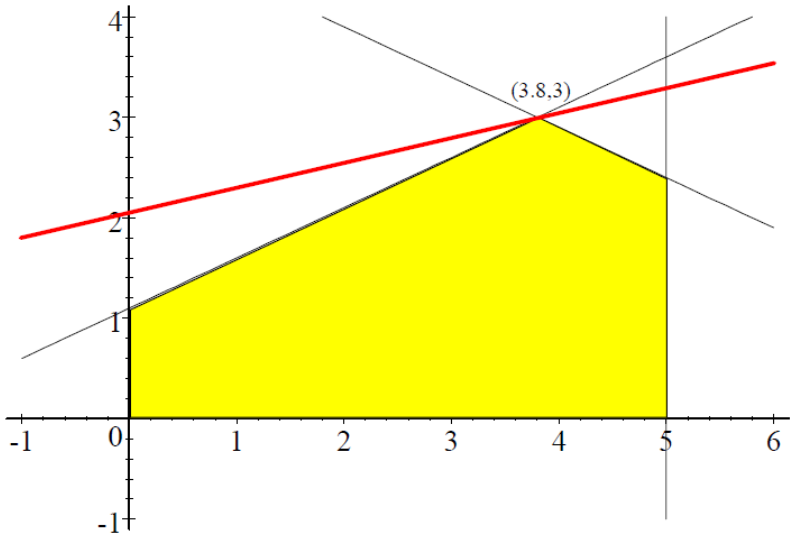


Figure 4.1: Illustration of the B&B method.

[1]The example is adapted from the lecture notes of Mustafa Celebi Pinar, available at https://www.ie.bilkent.edu.tr/~mustafap/courses/bb.pdf

Optimal solution of the relaxation is $(x_1, x_2) = (3.8, 3)$ with $z = 8.2$. Then we consider following two cases: Case 1: $x_1 \geq 4$; Case 2: $x_1 \leq 3$.

Observe that the set of feasible solutions is divided into two parts according to the different cases, see Fig. 4.2.

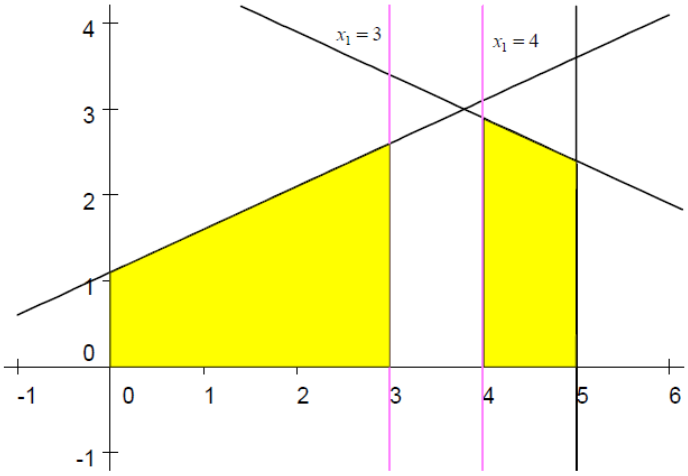


Figure 4.2: Illustration of the B&B method.

In Case 1, the relaxed LP

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & 4 \leq x_1 & \leq & 5 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

has optimal solution at $(4, 2.9)$ with $z = 7.6$ (see Fig. 4.3).

Now we should consider following two sub-cases: Case 1-1: $x_2 \geq 3$; Case 1-2: $x_2 \leq 2$.

In Case 1-1, the relaxed LP

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & 4 \leq x_1 & \leq & 5 \\ & & & x_2 & \geq & 3 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

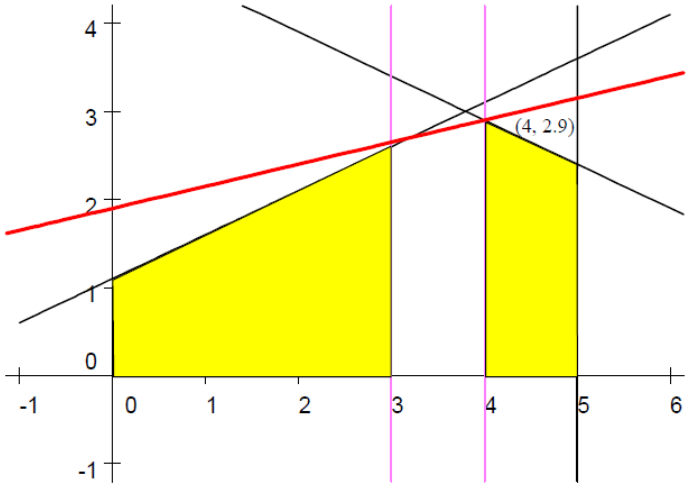


Figure 4.3: Illustration of the B&B method.

has no feasible solution (since $5x_1 + 10x_2 \geq 50$) so the IP has no feasible solution either.

In Case 1-2, the relaxed LP

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 5 \\ & & & x_1 & \geq & 4 \\ & & & x_2 & \leq & 2 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

has an optimal solution at $(4, 2)$ with $z = 4$. This is the optimal solution of the IP as well. Currently, the best value of z for the original IP is $z = 4$. (The corresponding figure is Fig. 4.4)

Now we should consider the branch of Case 2, when $x_1 \leq 3$. The relaxed LP

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 3 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

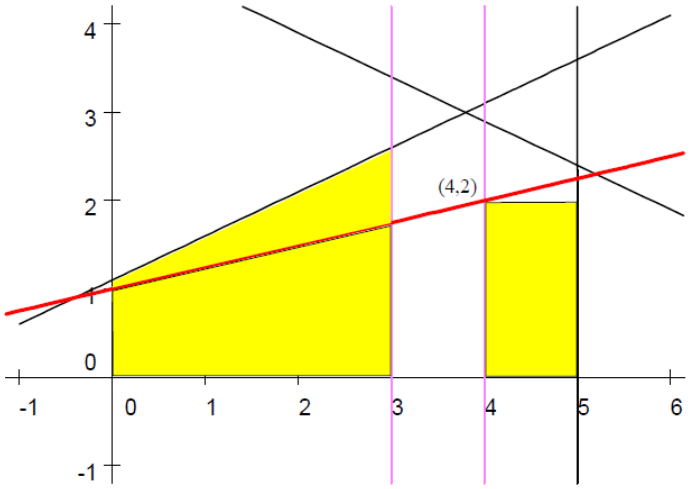


Figure 4.4: Illustration of the B&B method.

has an optimal solution at $(3, 2.6)$ with $z = 7.4$. We branch out further to two cases: Case 2-1: $x_2 \geq 3$ (not feasible!); Case 2-2: $x_2 \leq 2$ (see Fig. 4.5).

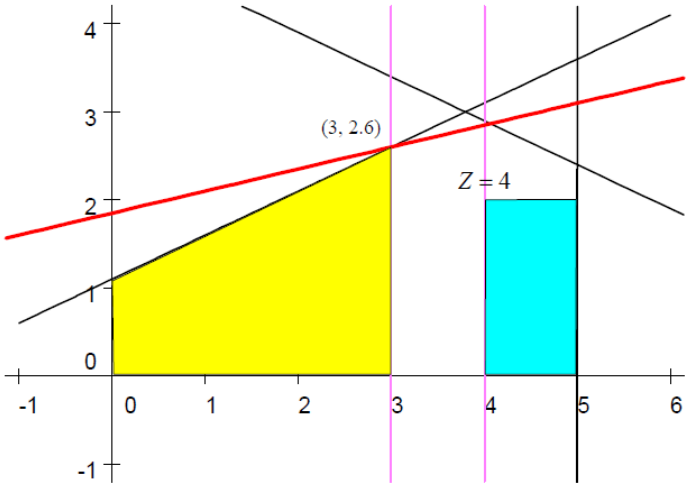


Figure 4.5: Illustration of the B&B method.

In Case 2-1, the relaxed LP

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 3 \\ & & & x_2 & \leq & 2 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

has an optimal solution at $(1.8, 2)$ with $z = 6.2$. In Case 2-2 the relaxed LP has no feasible solution, thus the IP has no solution either. We branch further with two cases: Case 2-1-1 $x_1 \geq 2$ and Case 2-1-2 $x_1 \leq 1$ (we still have the constraint $x_2 \leq 2$!; see Fig. 4.6).

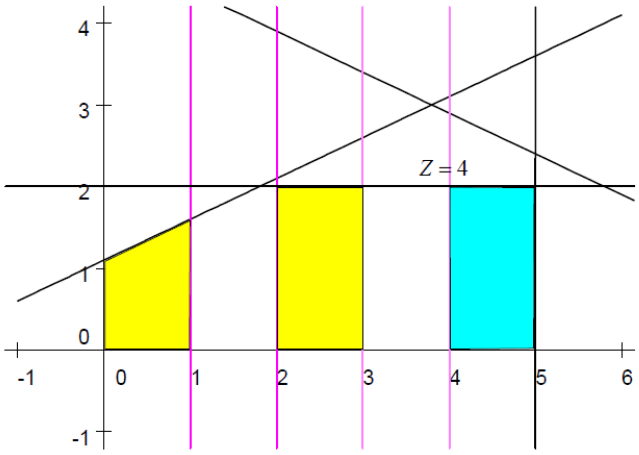


Figure 4.6: Illustration of the B&B method.

In Case 2-1-1 the LP relaxation

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & 2 \leq x_1 & \leq & 3 \\ & & & x_2 & \leq & 2 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

has an optimal at $(2, 2)$, with $z = 6$. Since this is better than the incumbent $z = 4$ at $(4, 2)$, this new integer solution is our current best solution.

In Case 2-1-2 the LP relaxation

$$\begin{array}{lrcrcl} \max z = & -x_1 & + & 4x_2 & & \\ \text{st} & -10x_1 & + & 20x_2 & \leq & 22 \\ & 5x_1 & + & 10x_2 & \leq & 49 \\ & & & x_1 & \leq & 1 \\ & & & x_2 & \leq & 2 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

has an optimal at $(1, 1.6)$ with $z = 5.4$. Then any integer solution in this region can not give us a solution with the value of z greater than 5.4. This branch is fathomed (see Fig. 4.7).

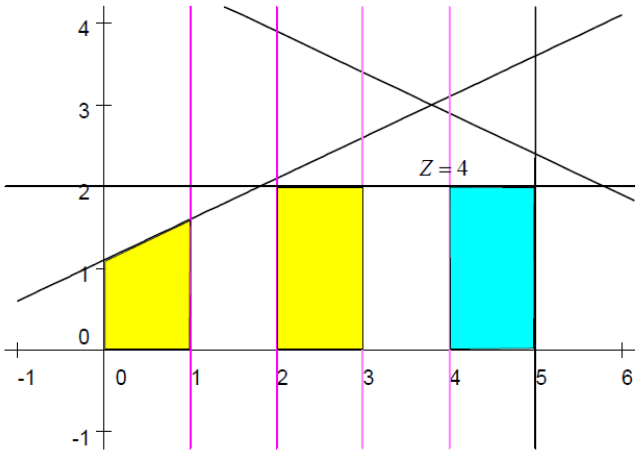


Figure 4.7: Illustration of the B&B method.

4.2.1 Rule of fathoming

A sub-problem is fathomed if

1. The relaxation of the sub-problem has an optimal solution with $z < z^*$ where z^* is the current best solution;

2. The relaxation of the sub-problem has no feasible solution;

3. The relaxation of the sub-problem has an optimal solution that has all integer values (or all binary if it is a 0-1 IP).

4.2.2 Branching tree

A display of all sub-problems that have been created is called a **tree**. Each sub-problem is referred to as a **node** of the tree, and each line connecting two nodes of the tree is called an **arc**. The constraints associated with any node of the tree are the constraints for the LP relaxation plus the constraints associated with the arcs leading from the original relaxed LP problem to the node. We will see an example for a branching tree in the following section.

Exercise. Construct the branching tree for the above example.

4.3 Knapsack problem

A **knapsack problem** is an IP with a single constraint.

Example: Josie Camper is going to a 2-day trip. There are 4 items that he wants to pack, but the total weight cannot exceeds 14kg. The following table shows the weight and utility of each item.

Item	Weight(kg)	Utility	Relative utility	Rank
1	5	8	1.6	1.
2	7	11	1.57	2.
3	4	6	1.5	3.
4	3	4	1.3	4.

The **Mathematical model** of the problem is the following. Let $x_i = 1$ if he brings item i and $x_i = 0$ if not. Then the task is to solve

$$\begin{array}{lrcrcrcrcl} \max & z & = & 8x_1 & + & 11x_2 & + & 6x_3 & + & 4x_4 & & \\ \text{st} & & & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 14 \\ & & & & & & & & & x_i & \in & \{0,1\} \quad (i = 1, \ldots, 4) \end{array}$$

The LP-relaxation can be solved easily. Put the items to the knapsack according to their relative utility (the most important first). If there is no more space for the item, put just an appropriate "part" of it. We put the first 2 items of 12 kg total, and "half" of the 3rd item: $x_1 = 1, x_2 = 1, x_3 = 0.5, z = 8 + 11 + 0.5 * 6 = 22$. Partitioning: $x_3 = 1$ (and we have 10kg free space) or $x_3 = 0$ (and we can use the rest of the item).

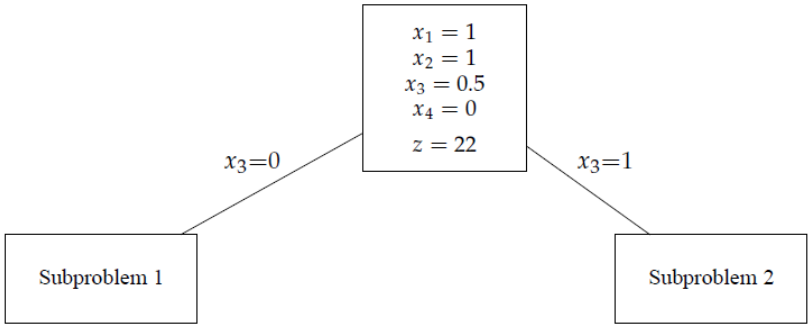


Figure 4.8: Branching tree for the knapsack problem

Sub-problem 1 is the following:

$$\begin{array}{lrcrcrcrcl} \max & z & = & 8x_1 & + & 11x_2 & + & 4x_4 & & \\ \text{st} & & & 5x_1 & + & 7x_2 & + & 3x_4 & \leq & 14 \\ & & & & & & & x_3 & = & 0 \end{array}$$

while **Sub-problem 2** is:

$$\begin{array}{lrcrcrcrcrcl} \max & z & = & 6 & + & 8x_1 & + & 11x_2 & + & 4x_4 & \\ \text{st} & & & & & 5x_1 & + & 7x_2 & + & 3x_4 & \leq & 10 \\ & & & & & & & & & x_3 & = & 1 \end{array}$$

Note that every sub-problem is characterized by the set of remaining objects (variables) and the total budget (the right-hand side). Observe that the above two sub-problems only differ in the value of budget (since we can ignore the constant term in the objective function). We can exploit this symmetry.

4.3.1 Dynamic programming for the knapsack problem

Instead of solving the problem just for 10 and 14, we solve the sub-problem for all meaningful values of the right-hand side (here for values $1, 2, ..., 14$). Having done that, we pick the best solution. This may seem unnecessary but surprisingly it can be faster. To make this work efficiently, we branch systematically on variables x_1, x_2, and so on.

In general, the knapsack problem is given by

$$\begin{array}{lrcrcrcrcrcl} \max & z & = & c_1x_1 & + & c_2x_2 & + & \ldots & + & c_nx_n & & \\ \text{st} & & & d_1x_1 & + & d_2x_2 & + & \ldots & + & d_nx_n & \leq & B \\ & & & & & & & & & x_i & \in & \{0,1\} \quad (i = 1, \ldots, n) \end{array}$$

For every $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, B\}$ we solve a subproblem with variables $x_1, \ldots, x_i$ and budget constraint k:

$$\begin{array}{lrcrcrcrcl} \max & z & = & c_1x_1 & + & c_2x_2 & + & \ldots & + & c_ix_i \\ \text{st} & & & d_1x_1 & + & d_2x_2 & + & \ldots & + & d_ix_i \quad \leq \quad k \\ & & & & & & & & & x_i \quad \in \quad \{0,1\} \quad (i = 1, \ldots, i) \end{array}$$

Let $f_i(k)$ be the value of the optimal solution. Then, the optimal solution to the original problem is $f_n(B)$. The optimal solution consisting of first i items either contains the ith item or it does not. If it does not contain the ith item then $f_i(k) = f_{i-1}(k)$. On the other hand, if the optimal solution using the first i items contains the ith item, then removing this item from the solution gives an optimal solution for first $i - 1$ items with budget $k - d_i$. We do not know which of the two situations happens, so we take the better of the two.

If from optimal solutions to smaller sub-problems we can build an optimal solution to a larger sub-problem we say that the problem has **optimal sub-structure.** A method that works based on this strategy called **dynamic programming**.

Based on the above considerations, he function $f_i(k)$ satisfies the following **recursion**:

$$f_i(k) = \begin{cases} 0, & i = 1 \\ \max\{f_{i-1}(k), c_i + f_{i-1}(k - d_i)\}, & i \geq 1 \end{cases}$$

We can calculate it by filling the table of all possible values. For example

$$f_3(10) = \max\{f_2(10), 6 + f_2(10 - 4)\} = \max\{11, 8 + 6\} = 14,$$

see bold numbers in the table.

$f_i(k)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	8	8	8	8	8	8	8	8	8	8
2	0	0	0	0	0	8	**8**	11	11	11	**11**	11	19	19	19
3	0	0	0	0	6	8	8	11	11	14	**14**	17	19	19	19
4	0	0	0	4	6	8	8	11	11	14	15	17	19	19	21

Dynamic programming characteristics

1. Problem can be divided into stages.

2. Each stage has a (finite) number of possible states, each associated a value.

3. Next stage only depends on the values of states of the previous stage.

4.4 Equivalent formulations

Finally in this chapter we show how practical solutions can be formulated as IP.

4.4.1 Fix-charge problems

Suppose activity i incurs a fixed charge if undertaken at any positive level. Let $x_i > 0$ be the level of activity (e.g. the number of units of production). In the model

- Let $y_1 = 1$ if activity i is undertaken at positive level ($x_i > 0$) and $y_1 = 0$ if $x_i = 0$.
- Constraint of the form $x_i \leq M_i y_i$ must be added to the formulation, where M_i must be large enough to ensure that x_i will be less than or equal to M_i.

4.4.2 Minimum level of production

If we produce the product i, then at least L must be produced. In the model

- Let $y_i = 1$ if we produce at least one i and $y_i = 0$ otherwise.
- Constraint of the form $x_i \geq L y_i$ must be added.

4.4.3 Either-Or constraint

Suppose we want to ensure that at least one of the following two constraints (and possibly both) are satisfied: $f(x_1, x_2, \ldots, x_n) \leq 0$ and $g(x_1, x_2, \ldots, x_n) \leq 0$. In the Model

- Let $y = 1$ if $g \leq 0$, and $y = 0$ ha $f \leq 0$.
- Add constraint $f(x_1, x_2, \ldots, x_n) \leq My$,
- add constraint $g(x_1, x_2, \ldots, x_n) \leq M(1 - y)$
- where $M \geq \max\{f, g\}$ for all $(x_1, x_2, \ldots, x_n)$.

4.4.4 If-Then constraint

Suppose we want to ensure that $f(x_1, x_2, \ldots, x_n) > 0$ implies $g(x_1, x_2, \ldots, x_n) \geq 0$. In the model

- Let $y = 1$ if $f > 0$, while $y = 0$ if $f \leq 0$.
- Add constraint $f(x_1, x_2, \ldots, x_n) \leq My$,
- add constraint $g(x_1, x_2, \ldots, x_n) \geq -M(1 - y)$
- where $M \geq \max\{f, g\}$ for all $(x_1, x_2, \ldots, x_n)$.

4.5 Exercises

4.5.1 The Telfa Corporation manufactures tables and chairs. A table requires 1 hour of labor and 9 square board feet of wood, and a chair requires 1 hour of labor and 5 square board feet of wood. Currently, 6 hours of labor and 45 square board feet of wood are available. Each table contributes \$8 to profit, and each chair contributes \$5 to profit. Formulate and solve an IP to maximize Telfa's profit.

4.5.2 (Capital Budgeting) Stockco is considering four investments. Investment 1 will yield a net present value (NPV) of \$16,000; investment 2, an NPV of \$22,000; investment 3, an NPV of \$12,000; and investment 4, an NPV of \$8,000. Each investment requires a certain cash outflow at the present time: investment 1, \$5,000; investment 2, \$7,000; investment 3, \$ 4,000; and investment 4, \$3,000. Currently, \$14,000 is available for investment. Formulate an IP whose solution will tell Stockco how to maximize the NPV obtained from investments 1-4. Modify the Stockco formulation to account for each of the following requirements:

1. Stockco can invest in at most two investments.
2. If Stockco invests in investment 2, they must also invest in investment 1.
3. If Stockco invests in investment 2, they cannot invest in investment 4.

4.5.3 (Facility location) There are six cities (cities 1-6) in county. The county must determine where to build fire stations. The county wants to build the minimum number of fire stations needed to ensure that at least one fire station is within 15 minutes (driving time) of each city. The times (in minutes) required to drive between the cities in the county are shown in Table 6. Formulate an IP that will tell Kilroy how many fire stations should be built and where they should be located.

	City 1	City 2	City 3	City 4	City 5	City
City 1	0	10	20	30	30	20
City 2		0	25	35	20	10
City 3			0	15	30	20
City 4				0	15	25
City 5					0	14
City 6						0

4.5.4 (IP with piecewise linear functions) Ewing Gas produces two types of gasoline (gas 1 and gas 2) from two types of oil (oil 1 and oil 2). Each gallon of gas 1 must contain at least 50 percent oil 1, and each gallon of gas 2 must contain at least 60 percent oil 1. Each gallon of gas 1 can be sold for 12 cent, and each gallon of gas 2 can be sold for 14 cent. Currently, 500 gallons of oil 1 and 1,000 gallons of oil 2 are available. As many as 1,500 more gallons of oil 1 can be purchased at the following prices: first 500 gallons, 25 cent per gallon; next 500 gallons, 20 cent per gallon; next 500 gallons, 15 cent per gallon. Formulate an IP that will maximize Ewing's profits (revenues – purchasing costs).

4.5.5 To graduate from CountryCollege University with a major in operations research, a student must complete at least two math courses, at least two OR courses, and at least two computer courses. Some courses can be used to fulfill more than one requirement: Calculus can fulfill the math requirement; operations research, math and OR requirements; data structures, computer and math requirements; business statistics, math and OR requirements; computer simulation, OR and computer requirements; introduction to computer programming, computer requirement; and forecasting, OR and math requirements. Some courses are prerequisites for others: Calculus is a prerequisite for business statistics; introduction to computer programming is a prerequisite for computer simulation and for data structures; and business statistics is a prerequisite for forecasting. Formulate an IP that minimizes the number of courses needed to satisfy the major requirements.

Chapter 5

Network problems

Network problems are widely spread in operations research, due to following main reasons:

1. They can be applied well to model real-life problems,
2. Efficient algorithms have been designed to solve them,
3. They have nice and elegant theoretical background.

In this section, at first we briefly discuss the basic elements of graph theory, then exhibit the most important graph algorithms, namely, the shortest path problem, the minimum spanning tree problem and the maximum flow problem. In Chapter 7 we will see the IP formulation of these problems, as well.

5.1 Graphs

Formally, an undirected (directed) **graph** (or **network**) $G = (V(G), E(G)) = (V, E)$ consists of two sets V and E, where $V \neq \emptyset$, while E is a set of (ordered) unordered pairs of elements of V. The elements of $V = \{1, 2, \ldots, n\}$ are called **nodes** (or vertices) and the elements of E are called **links** (or edges). A graph can mathematically be represented by its **adjacency matrix** $A = [a_{ij}]_{i,j=1,\ldots,n}$, which is an $n \times n$ matrix with entries $a_{ij} = 1$ if there is an edge (directed edge) between i and j and $a_{ij} = 0$ otherwise. For an undirected graph if the (i, j) edge exists, then $a_{ij} = a_{ji} = 1$, i.e. A is symmetric. If a function $w : E \to \mathbb{R}$ that assigns a real number w_{ij} to each edge (i, j) is given, then we say that the graph is **weighted**. Similarly one can introduce node weighting using a function $w : V \to \mathbb{R}$

For a graph G without self loops and multiple edges of n nodes[1] the number of links lies between 0 (empty graph) and $n(n-1)/2$ (complete graph).

The **degree** d_i of node i is the number links that are connected to i. If the network is directed, we can define the **in-degree** d_i^+ and **out-degree** d_i^- of a node i, these being the number of incoming links to u and the number of outgoing links from u, respectively. The weighted degree of a node can be calculated in a similar way using $w_i = \sum_u w_{ij}$ $(i = 1, \ldots, n)$, which is sometimes called the **strength** of i.

A **subgraph** $G' = (V', E')$ of $G = (V, E)$ is a graph where $V' \subseteq V$ and $E' \subseteq E$. If it contains all links of G that connects two nodes in V', it is said to be the **induced subgraph** by V'. A **clique** is a maximal complete subgraph of three or more nodes.

A **walk** $(i, k_1), (k_1, k_2), \ldots, (k_m, j)$ between two nodes i and j is an alternating sequence of nodes and edges, starting and ending at i and j, resp., in which each edge in the sequence is adjacent to its two endpoints. The length of the walk is the number of edges on it; for weighted graphs this is the the sum of wights of the edges on it. If all the nodes along the walk are distinct, then the walk is a **path**. The **shortest path** between i and j is a path between them where the length of the path is minimized. The (sub)graph is (strongly) **connected** if, for every pair of nodes i and j of the subgraph, there is a (directed) path from i to j. The connectedness is an equivalence relation, the equivalent classes called components (undirected) or strongly connected components (directed) of the graph.

A **bipartite graph** $G = (A, B, E)$ triple where $X, Y \subseteq V(G)$, $A \cap B = \emptyset$ and $E \subseteq A \times B$. Less formally, a bipartite graph is a graph whose nodes can be divided into two separate classes (A and B) and links connects nodes of different classes only. An $M \subseteq E(G)$ is a **matching** if any $u \in V(G)$ incident to at least one edge of M. M is *maximal*, if its size cannot be increased, and *complete*, if for all $u \in V$ there is an $e \in M$ such that u is incident to e.

5.2 Shortest path problem

One of the most important optimization problem on networks is finding shortest paths between vertices. It is widely-used, from GPS navigation and network communication to project management, layout design, robotics, computer graphics, among others. Often many problems reduce to finding shortest paths or use shortest paths as guidelines to optimal solutions.

[1] These graphs are called simple graphs.

Given a directed and weighted graph G with edge weights (lengths) $\ell(i,j)$. The length of a p path is $\ell(p)$ equals the sum of the length of edges that p contains. We also assume, that $\ell(i,i) = 0$ for all $(i,i) \in E(G)$, $i \in V(G)$. We draw the following to problems.

(1) Let s and t two distinguished vertices, find a **shortest path** p from s to t such that $\ell(p)$ is minimal.

(2) In general the task is to find **all shortest paths** from s to $v \neq s$.

Surprisingly all known algorithm that solves (1) also solves (2). Before presenting the solution we need to check when the shortest path exists. If there is *negative weight cycle* C ($\ell(C) < 0$) reachable from s then there is no solution to the problem (we can "go around" in this cycle many times to decrease the path length arbitrarily). On the other hand, if there is no such cycle, then the set of possible solutions is finite, and if it is not empty, then there is an optimal solution.

5.2.1 Bellman's algorithm

To solve problem (1) Bellman developed the **dynamic programming** method. Let $\ell(v)$ is the length of shortest path from s to v and $\ell_k(v)$ is the length of shortest path from s to v contains at most k edges. If the above defined paths do not exist, then let the value of $\ell(v)$ and $\ell_k(v)$ infinity. Let us use the function $p : V(G) \to V(G) \cup \{\emptyset\}$ to record these paths. Bellman's algorithm is the following.

1. Initially let $\overline{\ell}_0(s) := 0$, $p_0(s) := \emptyset$, $\overline{\ell}_0(v) := \infty$, $p_0(v) := \emptyset$, if $v \neq s$.

2. In step k let $\overline{\ell}_k(v) := \min\{\overline{\ell}_{k-1}(w) + \ell(w,v) : (w,v) \in E(G)\}$, and change the value of $p_{k-1}(v)$ to such w that achieves the minimum, if $\overline{\ell}_k(v) < \overline{\ell}_{k-1}(v)$.

Theorem 5.1 *The above algorithm correctly calculates the function ℓ_k therefore $\overline{\ell}_k \equiv \ell_k$. If there is no negative cycle in G then $\ell_{n-1}(v) = \ell(v)$ for all $v \in V(G)$, where $n = |V(G)|$. Moreover the vertices of a shortest $s - v$ path in reverse order are v, $p(v)$, $p^2(v)$, ..., $p^i(v) = s$, if $\ell(v) < \infty$, where $p = p_{n-1}$.*

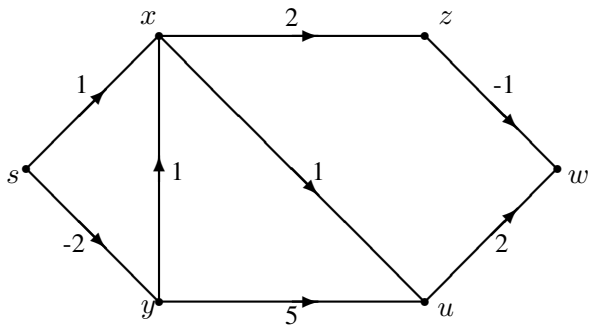
Proof. The first part of the statement can be proved by induction according to k. For $k = 0$ the statement is trivial. Suppose that for v the shortest $s - v$path contains at most $k+1$ is $(s, \ldots, w, v)$. From induction hypothesis it follows that there exist an $s - w$ path of length $\overline{\ell}_k(w) = \ell_k(w)$

contains at most k edges, thus $\ell_{k+1}(v) \geq \overline{\ell}_k(w) + l(w,v) \geq \overline{\ell}_{k+1}(v)$. Since $\ell_{k+1}(v) \leq \overline{\ell}_{k+1}(v)$, the proof is done.

If there is no negative cycle in G then any walk can be reduced to path of length not longer than of the walk. It follows that $\ell_{n-1} \equiv \ell_{n-1+i}$ for all $i \in \mathbb{N}$ and thus $\ell_{n-1} \equiv \ell$.

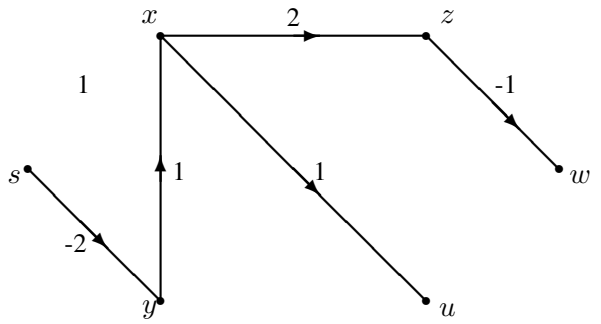
The role of p can also be proved by induction according to k. (Since $(p(v), p^2(v), \ldots, p^i(v) = s)$ is a shortest $s - p(v)$ path according to the assumption. Then adding $(p(v), v)$ edge to this path we obatin a shortest $s - v$ path.) □

Example: Let G be the following graph; then the steps of the algorithm can be summarized as follows.



	0		1		2		3		4		5	
	ℓ_0	p	ℓ_1	p	ℓ_2	p	ℓ_3	p	ℓ_4	p	ℓ_5	p
s	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$	0	$\emptyset$
x	∞	$\emptyset$	1	s	-1	y	-1	y	-1	y	-1	y
y	∞	$\emptyset$	-2	s	-2	s	-2	s	-2	s	-2	s
z	∞	$\emptyset$	∞	$\emptyset$	3	x	1	x	1	x	1	x
u	∞	$\emptyset$	∞	$\emptyset$	2	x	0	x	0	x	0	x
w	∞	$\emptyset$	∞	$\emptyset$	∞	$\emptyset$	2	z	0	z	0	z

The table is filled column by column. Initially $\ell_0(s) = 0$, $\ell_0(v) = \infty$ for all $v \neq s$, while $p = \emptyset$. Then ℓ_k and p_k is calculated via recursion for $k \geq 1$. The last column of the table contains not only the length of shortest $s - v$ paths ($v \in G$), but the edges of the path can be read there. If we depict only the edges contained in the shortest path, these are the values of function p, then we get the so-called **shortest-path tree**. This is **rooted tree** with root s, each vertex is reachable from s, means that $d_-(s)$ = 0 and $d_-(v)$ = 1, if $s \neq v \in V(G)$.



We can easily get the all shortest $s - v$ paths and their length:

1. $s - x$ path: (s, y, x), length: $d(x) = -1$
2. $s - y$ path: (s, y), length: $d(y) = -2$
3. $s - u$ path: (s, y, x, u), length: $d(u) = 0$
4. $s - z$ path: (s, y, x, z), length: $d(z) = 1$
5. $s - w$ path: (s, y, x, z, w) length: $d(w) = 0$

Remarks:

1. The method can be applied in case when G contains negative cycle C. In this case, the existence of a negative cycle is shown by $v = p^i(v)$ for some $v \in V(G)$ and $i \in \mathbb{N}$, moreover p can be used to determine C. Also it can be decided wheter a given vertex v is on negative walk or not. Add a new vertex q to the graph and edges (q, v), (v, q) with zero weight. It is clear that $\ell_n(q) < 0$ if and only if v is on the negative walk.

2. The connectedness of an undirected graph G can be tested with the algorithm. Add directed edges (i, j) and (j, i) instead of the undirected (i, j) with weight one. If all $s-v$ shortest paths has finite length then G is connected; if $\ell(v) = \infty$, then s and v are in different components.

3. The main substance of the algorithm is that all $v \in V(G)$ there is a $w \in V(G)$, such that $\ell_k(v) = \ell_{k-1}(w) + \ell(w, v)$. It roughly means that there is connection between the optimal structures: the "larger" can be obtained by the "smaller" one. Such observation can often be used to design efficient algorithms. The method is called **dynamic programming**.

5.2.2 Dijkstra's algorithm

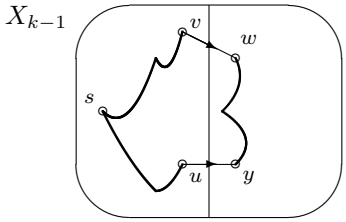
Given a directed graph G, assume that the weight $\ell(v, w)$ of each edge is non-negative. Since there is no negative cycle in G, therefore there is a shortest path from s to v (or it is infinity if there is no path from s to v). The shortest path starting from a node s and the shortest-path tree can be obtained by $n = |V(G)|$ iteration using Dijkstra's algorithm published in 1956. The steps of the algorithm are the following.

1. Let $X_1 := \{s\}$, $d(s) = 0$, $d(v) = \infty$ if $v \neq s$ and $T_1 := \{s\}$, a single vertex tree.

k. Assume that X_{k-1} and $d(v)$ ($v \in X_{k-1}$) are defined already. Choose a $w \in V(G) \backslash X_{k-1}$ such that $d(v)+\ell(v, w)$ minimal, where $v \in X_{k-1}$. Let $X_k := X_{k-1} \cup \{w\}$, $d(w) := d(v)+\ell(v, w)$, and T_k is the tree obtained by T_{k-1} by adding w and (v, w) path.

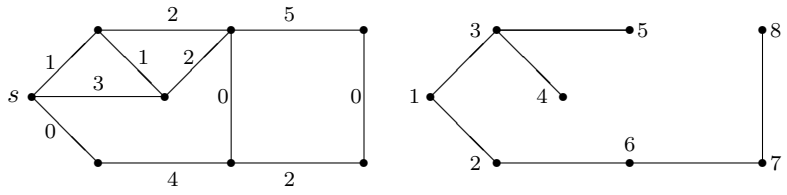
Theorem 5.2 *Using Dijkstra's algorithm, after n steps the value $d(v)$ is the length of the shortest $s - v$ path for all $v \in V(G)$. Furthermore T_n is the s-rooted tree contains all shortest paths from s.*

Proof. We prove by induction according to the iteration steps. The induction hypothesis says that if $v \in X_{k-1}$ then $d(v)$ is the length of the shortest $s - v$ paths; assume that for $w \in X_k \backslash X_{k-1}$, $d(w) = \min_{v \in X_{k-1}} d(v) + \ell(v, w)$ calculates the length of the shortest $s - w$ path. Then the p shortest $s - w$ path leaves X_{k-1} from u as its final node. Let y is the upcoming node in p after u (it can be assumed that $y \neq w$).

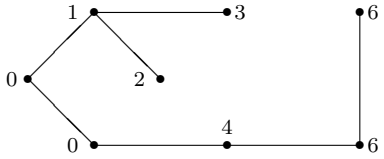


Since the length of p is smaller than $d(w)$, the length of $s - y$ path contained in p is also smaller than $d(w)$. (We used the non-negativity of weights here!) But then the algorithm had to choose y in step k, that is a contradiction. The proof of the role T_n is also done by induction. Thus T_{k-1} stores the shortest $s - v$ paths if $v \in X_{k-1}$; T_k is obtained by adding such (v, w) where w is reachable by a length $d(w)$ path from s. □

Example. On the left-hand side given undirected weighted graph G with distinguished node s. On the right-hand side there is the shortest-path tree with root s; the numbering indicates that in which step a node added to the tree.

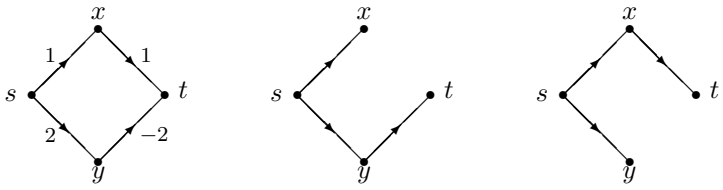


The length of the shortest paths can be read easily, by using $d(s) = 0$ and for $v \neq s$ simply sum up the edges weights on the unique $s - v$ path on the tree.



Remark. Dijkstra's algorithm is simple and faster than Bellman's, but it can be applied only if the weights are non-negative.

Example. On the left-hand side the given graph, in the middle is the correct shortest-path tree obtained by Bellman's algorithm, on the right-hand side the tree obtained by using Dijkstra's algorithm.

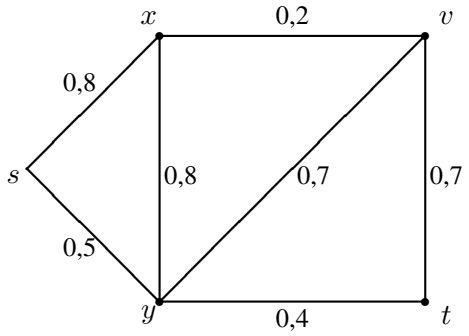


Observe that Dijkstra's method does not find the shortest $s - t$ paths, since $d(t) = 0$, but $d(t) \neq 2$ and $s - t$ path is not $\{s, x, t\}$ but $\{s, y, t\}$.

5.2.3 An application: largest probability path

Suppose that we are given a directed graph. Each edge e is assigned a number $p_e \in [0, 1]$ represents the passage probability (i.e. the probability that the edges is usable/open) on it. The probabilities are fixed in advance and independent from each other. Given two nodes, s and t, we are finding the $s - t$ path of the largest probability.

Example.



If we pick some edges, the probability that all can be used is the product of the respective probabilities due to the independence condition. In the example above, the highest probability $s - t$ path is (s, x, y, v, t) and the probability is $0,8 \times 0,8 \times 0,7 \times 0,7 = 0,3156$. In general we can do the following: change p_e to $-\ln p(e)$ for each edge e and find the shortest $s - t$ path with respect to this weighting. Let R and Q be the edge sets of two different $s - t$ paths. Then the probability of usability of these paths are

$$\prod_{e \in R} p_e \text{ and } \prod_{e \in Q} p_e,$$

respectively, while their length (with respect to the transformation) are

$$-\sum_{e \in R} \ln p_e \text{ and } -\sum_{e \in Q} \ln p_e.$$

If

$$-\sum_{e \in R} \ln p_e < -\sum_{e \in Q} \ln p_e, \text{ then } \sum_{e \in R} \ln p_e > \sum_{e \in Q} \ln p_e.$$

From this follows that

$$\ln \prod_{e \in R} p_e > \ln \prod_{e \in Q} p_e,$$

and due to the monotonicity of the logarithm function we get the

$$\prod_{e \in R} p_e > \prod_{e \in Q} p_e$$

inequality, which proves that finding the shortest path (using any of the previously discussed methods) of the transformed graph is the solution to the largest probability path problem.

5.3 Minimum spanning tree

The minimum spanning tree problem also has a rich history and it has numerous applications too. Let us start with an example. A power company delivers electricity from its power plant to neighboring cities. The cities are interconnected by power lines operated by various operators. The power company wants to rent power lines in the grid of least total cost that will allow it to send electricity from its power plant to all cities. An optimization problem can phrased in mathematical terms as follows.

Given a weighted network $G = (V, E)$, the problem is to find a set of edges $F \subseteq E$ where $w(F) = \sum_{e \in F} w(e)$ is minimal so that (V, F) is a tree. We say that (V, F) is a **spanning tree** because it spans all vertices.

5.3.1 Kruskal's and Prim's algorithms

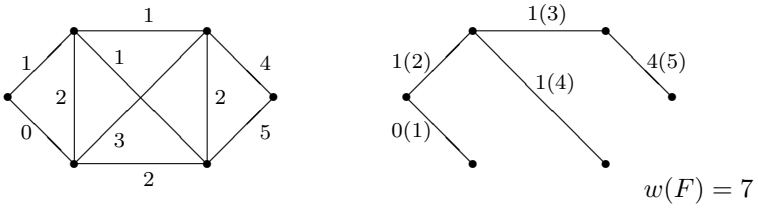
Kruskal (1956) and Prim (1957) independently developed algorithms fro the problem having almost linear-time implementations.[2] The two algorithms are very similar, the difference can be captured only in one step. **Kruskal's algorithm** is the following.

1. Let $F = \emptyset$ and all edges initially *unprocessed*.
2. Find an unprocessed edge e of smallest weight $w(e)$.
3. If $(V, F \cup \{e\})$ is a forest, then add e to F. Mark e as *processed* and repeat until all edges have been processed.
4. Report (V, F) as a minimum spanning tree.

[2] We should note here that the first know algorithm for the problem was discovered by Borůvka in 1926.

Prim's algorithm changes 1. to 1.': Find an unprocessed edge e of smallest weight $w(e)$ that shares an endpoint with some edge in F.

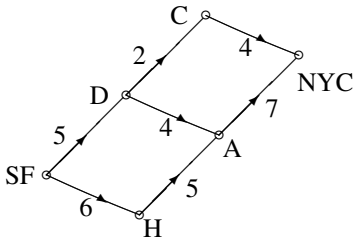
Example. On the left hand-side given the weighted graph G, on the right-hand side its minimum spanning. In brackets the number indicates the step number when the edge added to the tree using Kruskal's algorithm.



5.4 Maximum flow problem

Let us start with a classic example. An Small Airlines runs a network between the major US cities. In each route (between two cities) the maximum number of flights are given (per day, indicated by the numbers in the table). The management is interested in the maximum number of passengers (i.e. number of flights) from San Francisco to New York on a daily basis.

Source	Target	Trucks
San Francisco	Denver	5
San Francisco	Houston	6
Denver	Atlanta	4
Denver	Chicago	2
Houston	Atlanta	5
Atlanta	New York	7
Chicago	New York	4



From the table we can readily define a directed graph; the weight of an edge represents the number of flights run between the two represented cities per day. We need to formalize the following rules on the delivery plan:

1. If k flights run on route (v, w) k is equivalent to $-k$ flights run on route (w, v).

2. The number of flights run in a route cannot exceeds the maximum number of flights can run.

3. Except SF and NYC, the incoming and outgoing flights of each city should be equal (i.e. there sum is zero using 1.).

Let G be a directed graph with two special points: s is called **source** and t is called **target (or sink)**. Let $c(v, w)$ be the **capacity** of (v, w), that is the maximum number of flights on that edge. Furthermore, let $c(v, w) = 0$, if (v, w) is not an edge in G. The function $f : V(G) \times V(G) \to \mathbb{R}$ is a **flow** if it satisfies the following 3 conditions:

(1) **Skew symmetry.** For all v, w points $f(v, w) = -f(w, v)$. If $f(v, w) > 0$, we say that *there is* a flow from v to w.

(2) **Capacity constraint.** For all v, w points $f(v, w) \leq c(v, w)$. If for (v, w) $f(v, w) = c(v, w)$ holds, we say that the flow *saturates* (v, w).

(3) **Flow conservation rule.** Except s and t, for each v point $\sum_{w \in V(G)} f(v, w) = 0$ holds.

The **value** of the flow is $|f| := \sum_{w \in V(G)} f(s, w)$, i.e. the quantity leaves the source. The **maximum flow problem** is the finding the maximum value flow. The central concept of the investigations is the **cut**. An $X, \overline{X}$ cut is a partition of $V(G)$ such that any node is either $s \in X$ or $t \in \overline{X} = V(G) \backslash X$. The **capacity** of cut $X, \overline{X}$ is $c(X, \overline{X}) := \sum_{v \in X, w \in \overline{X}} c(v, w)$. A cut with the minimum capacity is called **minimum cut** If f is a flow, $X, \overline{X}$ is a cut, then the *flow goes through on the cut* is $f(X, \overline{X}) := \sum_{v \in X, w \in \overline{X}} f(v, w)$.

Lemma 5.1 *For each flow f and cut $X, \overline{X}$ flow goes through on the cut equals to the value of the flow.*

Proof.

$$f(X, \overline{X}) = \sum_{v \in X, w \in \overline{X}} f(v, w) = \sum_{v \in X, w \in V(G)} f(v, w) - \sum_{v \in X, w \in X} f(v, w) = |f|,$$

since $\sum_{v \in X, w \in V(G)} f(v, w) = |f|$ due the conservation rule and $\sum_{v \in X, w \in X} f(v, w)$ is zero due to skew symmetry □

This means that the value of the flow equals to the quantity that leaves X. Since $f(v, w) \leq c(v, w)$, thus

$$|f| = \sum_{v \in X, w \in X} f(v, w) \leq \sum_{v \in X, w \in \overline{X}} c(v, w)$$

for each f flow and $X, \overline{X}$ cut. Therefore the value of the maximum flow does not exceeds the value of the minimum cut. It is more surprising that the two values are equal. In fact, this is a special case of the strong duality theorem, and we are going to prove it after introducing some further notions.

The **residual capacity** of a flow f between nodes v and w is $r(v,w) := c(v,w) - f(v,w)$. In fact we can increase the value of the flow by $r(v,w)$ from v to w (by decreasing the $f(w,v)$ by the same value). The R **residual graph** of a flow is graph where $V(R) = V(G)$ and (v,w) is an edge if $r(v,w) > 0$. Note that it may contain edges that are not in G. A p is called an **augmenting path** for flow f if it is a path from s to t in R. The residual capacity of p, $r(p)$, is an edge weight of p with a minimum capacity. The "augmenting nature" of p is straightforward. Increasing the flow by $r(p)$ in each edge of p the value of the flow is increased by $r(p)$. (Note that if $f(v,w)$ is changing, $f(w,v)$ is changing too!)

Lemma 5.2 *Let f be an arbitrary and f^* be a maximal flow of G, respectively. If R is the residual graph of f, then the value of the maximum flow of R is $|f^*| - |f|$.*

Proof. Let f' be an arbitrary flow on R and define the $f + f'$ flow as $(f + f')(v,w) := f(v,w) + f'(v,w)$. Then $f + f'$ is a flow on G and its value is $|f| + |f'| \leq |f^*|$, thus $|f'| \leq |f^*| - |f|$. Similarly define $f^* - f$ as $(f^* - f)(v,w) := f^*(v,w) - f(v,w)$ on R and its value is $|f^*| - |f|$, thus it is a maximum flow on R. □

Theorem 5.3 (Max-Flow-Min-Cut) *The following statements are equivalent:*

(i) The flow f is maximal.

(ii) There are no augmenting paths to f.

(iii) There is a cut $X, \overline{X}$ such that $|f| = c(X, \overline{X})$.

Proof. (i)⇒ (ii) If there is a p augmenting path to f, then increase the value of the flow through the edges of p. (ii)⇒ (iii) Suppose that there is no augmenting path to f. Let X be the set of points that are reachable from s using the edges of R and let $\overline{X} = V(G) \backslash X$. Then $X, \overline{X}$ is a cut ($s \in X$, $t \in \overline{X}$), and

$$|f| = \sum_{v \in X, w \in \overline{X}} f(v,w) = \sum_{v \in X, w \in \overline{X}} c(v,w) = c(X, \overline{X}),$$

since $v \in X$, $w \in \overline{X}$ follows that (v,w) is not an edge of R, thus $f(v,w) = c(v,w)$. (iii)⇒ (i) Since $|f| \leq c(X, \overline{X})$ for each flow f and $X, \overline{X}$ cut, from $|f| = c(X, \overline{X})$ follows that f is a

maximum flow and $X, \overline{X}$ is a minimum cut. □

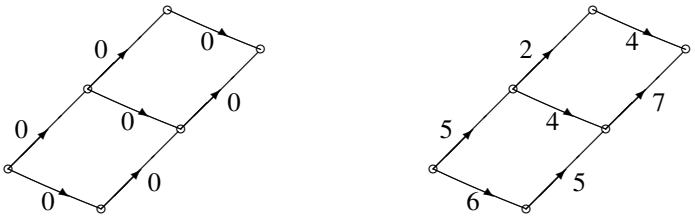
The above theorem is the basis of the **augmenting path method** of Ford and Fulkerson. It starts from a zero flow ($f(v, w) = 0$ for each (v, w) edge), and then repeats the augmenting step until arrives to a flow, in which there is no augmenting path. The **augmenting step** is following: Find a p augmenting path of the current flow. Increase the value of f by $r(p)$ on the edges of p.

Theorem 5.4 (Integrality) *If the capacities are integer values in the maximum flow problem, then there is an integer valued maximum flow.*

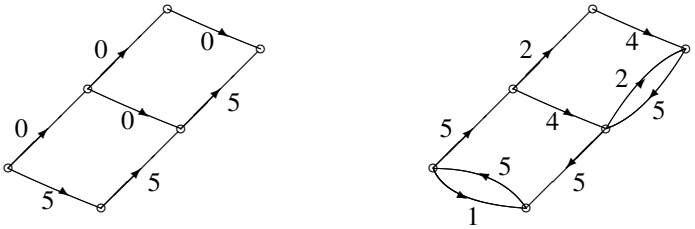
Proof. Assume that the capacities are integer values. Then the augmenting path method increases the value of the flow at least by one, thus f^* maximum flow is obtained at most $|f^*|$ steps. On the other hand the initial integer value (zero) is changed by an integer number in every step, thus $f^*(v, w)$ is integer for all (v, w) edge. □

In the following example we illustrate the Ford-Fulkerson method. On the left side, we show the actual value of the flow, on the right side there is the corresponding residual graph with the augmenting path.

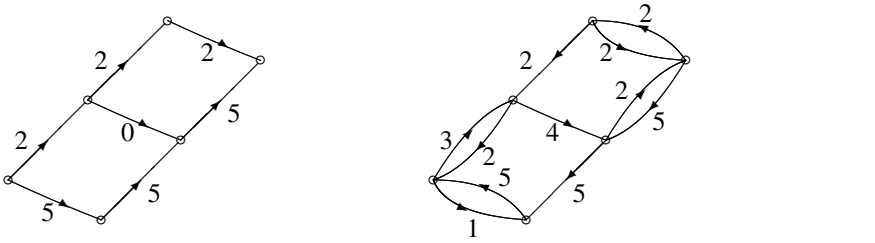
Initialization. The augmenting path: SF-H-A-NYC, $r(p) = 5$



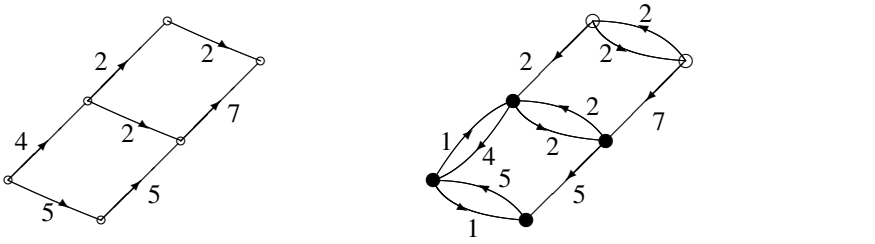
1st step. The augmenting path: SF-D-C-NYC, $r(p) = 2$



2nd step. The augmenting path: SF-D-A-NYC, $r(p) = 2$

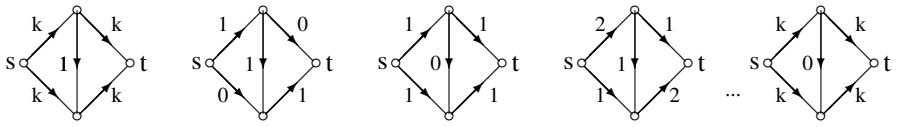


3rd step. There are no augmenting paths, $|f^* = 9|$, X, $\bar{X}$ (full and empty circles on the figure)



Remark: The augmenting path method does not necessarily converge, moreover the number of steps can be large even in case of integer numbers on the edges. However, Edmonds and Karp showed in 1969, that selecting an augmenting path with a *minimum number of edges* the running time is cnm^2 steps, where c is a constant, n is the number of nodes and m is the number of edges in G

Example: The first figure shows the capacities, the rest shows the possible solutions if the augmenting path uses the vertical edge. (We show just the process, and do not the residual graph). Then the number of steps is k. If we chose the shortest augmenting path in every step, then we reach the optimum in two steps.



5.5 Exercises

5.5.1 Show that if the edge weights of a graph are all different then the minimal spanning tree is unique.

5.5.2 We say that (e, e') directed edges are *anti-parallel* if their endpoints are the same but their direction is different. Show that in any flow network there is a maximal flow that does not uses at least one of e and e'.

5.5.3 Suppose that there are more than one source and more than one sink in a low network. Show that how the Ford-Fulkerson algorithm can be used in that case.

5.5.4 Suppose that in a flow network each edge capacity is even. Show that then the maximum flow is also even.

5.5.5 Given a directed graph G and two distinguished nodes s and t. Design an algorithm that counts the number of edge-disjoint $s - t$ paths.

5.5.6 Given a directed graph G and two distinguished nodes s and t. Design an algorithm that counts how many edges should be deleted make t non-reachable from s.

5.5.7 Given a directed graph G and two distinguished nodes s and t. Design an algorithm that counts the number of node-disjoint $s - t$ paths.

Chapter 6

Matrix formulation and totally unimodular matrices

We shall see that many graph problems can be formulated as a linear (LP), or integer programming (IP) problem. The difficulty of IP is that polyhedrons contains the feasible solutions may not have corner points of integer coordinates. However, if we know that the corner points have integer coordinates then solving the relaxed problem (using the simplex algorithm) we obtain the integer optimal solution immediately. Hoffman and Kruskal proved in 1956 that this condition holds for an important class of problems. Their result is of central importance in the study of network problems.

In this section, first we give a brief overview of the basics of linear algebra. Then define totally unimodularity of matrices and discuss some important results of TU matrices and their connection to LP.

6.1 Linear algebra overview

We start with the very basic definitions and notations we will use.

- **scalar** = a number; can be real ($\pi = 3.14\ldots$), rational ($3/4$), integer (5, -8), etc.
- **vector** = sequence of numbers, for example $(3, 1, 0, 2)$, we often write $x = \begin{bmatrix} 3 & 2 & 0 & 1 \end{bmatrix}$ **raw vector**, or $x = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 2 \end{bmatrix}$ **column vector**

- **multiplying** $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ by a :

$$ax = \begin{bmatrix} ax_1 & ax_2 & \cdots & ax_n \end{bmatrix}$$

- **addition** of $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ and $y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}$ vectors (of the same size):

$$x + y = \begin{bmatrix} x_1 + y_1 & x_2 + y_2 & \dots & x_n + y_n \end{bmatrix}$$

- **scalar product** of $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ and $y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}$ vectors (of the same size) :

$$xy = x_1y_1 + x_2y_2 \cdots + x_ny_n$$

- x and y **orthogonal**, if $xy = 0$.

- **matrix** = 2-dimensional array of numbers, for example

$$A = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix}$$

- in general, an $m \times n$ matrix is A with entry a_{ij} in the i-th row and j-th column. The i-th row is $\begin{bmatrix} a_{i1}, a_{i2}, \dots, a_{im} \end{bmatrix}$, while the j-th column is $\begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{bmatrix}$

- **multiplying a matrix by a scalar**:

$$2 \cdot A = 2 \cdot \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 & 2 \\ 6 & 4 & 8 & 0 \\ 4 & 6 & 0 & 2 \\ 0 & 8 & 2 & 4 \end{bmatrix}$$

- **adding matrices** of the same size

$$A + B = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 1 & 0 & 2 \\ 1 & 2 & 5 & 1 \\ 6 & 0 & 1 & 0 \\ 3 & 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 1 & 3 & 3 \\ 4 & 4 & 9 & 1 \\ 8 & 3 & 1 & 1 \\ 3 & 5 & 3 & 5 \end{bmatrix}$$

- **multiplying matrices**: A of size $n \times m$-es multiplied by B of $m \times k$

$$A \cdot B = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 & 4 \\ 2 & 0 & 1 \\ 0 & 2 & 1 \\ 3 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 6 & 7 & 10 \\ 13 & 11 & 18 \\ 15 & 2 & 14 \end{bmatrix}$$

  - Note that $AB \neq BA$ generally
  - except for this, matrix addition and multiplication obey similar laws as numbers
  - from now on vector with m entries is to be treated as a $m \times 1$ matrix (column)

- multiplying matrix by a vector is just like multiplying two matrix

- **transpose** of a matrix:

$$A = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \end{bmatrix}, A^T = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 3 \\ 3 & 4 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

  - Note that $(A^T)^T = A$ and $(AB)^T = B^T A^T$

- **inverse matrix** of a A is matrix A^{-1} such that $A^{-1} \cdot A = A \cdot A^{-1} = I$ (if such A^{-1} matrix exists), where is is the identity matrix[1]

- A **system of linear equations** has the following form

$$\begin{bmatrix} 1 & 0 & 3 & 1 \\ 3 & 2 & 4 & 0 \\ 2 & 3 & 0 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \\ 0 \end{bmatrix}$$

  - Using the matrix notation we can simply write it as $Ax = b$.

[1] Its diagonal elements are 1, its non-diagonal elements are 0.

6.2 Simplex algorithm in matrix form

We are given a **LP in standard form**:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad i = 1, 2, \ldots, m$$

$$x_j \geq 0 \qquad j = 1, 2, \ldots, n$$

$$\max \sum_{j=1}^{n} c_j x_j$$

Adding the non-negative artificial variables we get

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i \qquad i = 1, 2, \ldots, m$$

$$x_j \geq 0 \qquad j = 1, 2, \ldots, n+m$$

$$\max \sum_{j=1}^{n} c_j x_j$$

In **matrix form** we can write as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & & \\ a_{21} & a_{22} & \cdots & a_{2n} & & 1 & \\ & & \vdots & & & & \ddots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{n+m} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_n & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{n+m} \end{bmatrix} = z$$

Matrices and vectors used above:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & & \\ a_{21} & a_{22} & \cdots & a_{2n} & & 1 & \\ & & \vdots & & & & \ddots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & & & 1 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{bmatrix} \qquad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

The problem in **matrix form** is written as

$$\begin{aligned} \max \quad c^T x &= z \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

We know that a dictionary is uniquely determined by its basic variables. Let $\mathcal{B}$ be the index set of the basic variables, $\mathcal{N}$ be the index set of the non-basic variables. Divide the matrices and vectors to two parts, based on the role (basic or non-basic) of their elements in the dictionary.

Let B denote the **basis** matrix formed by taking the columns of A corresponding to the basic variables $x_{\mathcal{B}}$. Let N denote the columns of A corresponding to the non-basic variables in $x_{\mathcal{N}}$. Divide the vectors as

$$c = \begin{bmatrix} c_{\mathcal{B}} \\ c_{\mathcal{N}} \end{bmatrix} \text{ and } x = \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{bmatrix}.$$

We obtain that

$$Ax = \begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{bmatrix} = Bx_{\mathcal{B}} + Nx_{\mathcal{N}},$$

$$c^T x = \begin{bmatrix} c_{\mathcal{B}} & c_{\mathcal{N}} \end{bmatrix} \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{bmatrix} = c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{N}}^T x_{\mathcal{N}}$$

and hence the optimization problem is given in the form:

$$\begin{aligned} Bx_{\mathcal{B}} + Nx_{\mathcal{N}} &= b \\ x &\geq 0 \\ \hline \max c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{N}}^T x_{\mathcal{N}} \end{aligned}$$

Assuming that B **is invertible**, we can rewrite:

$$\begin{aligned} Ax &= b \\ Bx_{\mathcal{B}} + Nx_{\mathcal{N}} &= b \\ B^{-1}(Bx_{\mathcal{B}} + Nx_{\mathcal{N}}) &= B^{-1}b \\ B^{-1}Bx_{\mathcal{B}} + B^{-1}Nx_{\mathcal{N}} &= B^{-1}b \\ x_{\mathcal{B}} + B^{-1}Nx_{\mathcal{N}} &= B^{-1}b \\ x_{\mathcal{B}} &= B^{-1}b - B^{-1}Nx_{\mathcal{N}} \end{aligned}$$

Now we can substitute $x_{\mathcal{B}}$ to the objective function

$$\begin{aligned} z = c^T x &= c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{N}}^T x_{\mathcal{N}} \\ &= c_{\mathcal{B}}^T (B^{-1}b - B^{-1}Nx_{\mathcal{N}}) + c_{\mathcal{N}}^T x_{\mathcal{N}} \\ &= c_{\mathcal{B}}^T B^{-1}b + (c_{\mathcal{N}}^T - c_{\mathcal{B}}^T B^{-1}N)x_{\mathcal{N}} \end{aligned}$$

We put it together to obtain the corresponding dictionary:

$$\begin{aligned} x_{\mathcal{B}} &= B^{-1}b - B^{-1}Nx_{\mathcal{N}} \\ \hline z &= c_{\mathcal{B}}^T B^{-1}b + (c_{\mathcal{N}}^T - c_{\mathcal{B}}^T B^{-1}N)x_{\mathcal{N}} \end{aligned}$$

The **basic solution**, when $x_{\mathcal{N}} = 0$, is given as $x_{\mathcal{B}} = B^{-1}b$, with the value of the objective function $z = c_{\mathcal{B}}^T B^{-1}b$. **The solution is optimal (maximal) if** $c_{\mathcal{N}}^T - c_{\mathcal{B}}^T B^{-1}N \leq 0$, means that the constant coefficients of the non-basic variables are non-positive.

6.3 TU matrices

A matrix A is **totally unimodular** (TU) if its every square submatrix has determinant 1, -1, or 0. (It follows that a TU matrix has only 0, $+1$ or -1 entries)

Theorem 6.1 (Hoffman-Kruskal) *If A is totally unimodular and all entries of b and c vectors are integers, then the basic solutions of the $\max c^T x$ s.t. $Ax \leq b$ linear program are integers. In other words, the coordinates of the corner points of the $P = \{x : Ax \leq b\}$ polyhedron are integers.*

Proof. The dictionary for $x_{\mathcal{B}}$ basic solution is given in the form

$$x_{\mathcal{B}} = B^{-1}b - B^{-1}Nx_{\mathcal{N}}$$

$$\overline{z = c_{\mathcal{B}}^T B^{-1}b + (c_{\mathcal{N}}^T - c_{\mathcal{B}}^T B^{-1}N)x_{\mathcal{N}}}$$

We can calculate the entries of B^{-1} using the ***Cramer-rule***, such that

$$(B^{-1})_{i,j} = \frac{(-1)^{i+j}\det(B^{ji})}{\det(B)},$$

where B^{ij} is the matrix obtained from B by omitting raw j and column i. Since now A is TU and $\det(B) \neq 0$, then $\det(B) = \pm 1$. It is easy to see that each entry of B^{ji} is integer. Since $x_{\mathcal{B}} = B^{-1}b$ it immediately follows that $x_{\mathcal{B}}$ is an integer vector. □

Remark. Obviously, if we solve the problem using the simplex method, then the other elements of the dictionary (namely $c_{\mathcal{B}}^T B^{-1}b$ and $(c_{\mathcal{N}}^T - c_{\mathcal{B}}^T B^{-1}N)x_{\mathcal{N}}$) are also integers and no numerical errors occur.

The following theorem is essential for solving network problems (will be discussed in the next chapter) formulated as linear programs.

Theorem 6.2 *An incidence matrix A of a directed graph G is totally unimodular.*

Proof. We use induction according to the size of sub-determinants. The 1×1 sub-matrices are ± 1 or 0, thus the statement holds for $k = 1$. If B is a $(k+1) \times (k+1)$ sub-matrix, then there are two cases. If B has a column with at most one non-zero element, then calculate the determinant expressed from this column. Due to the induction hypothesis this is 0 or ± 1. If all column of B contains exactly two non-zero elements (i.e. $+1$ and -1), then summing all the rows of B we get the zero vector, means that the rows of B linear dependent. Then $det(B) = 0$, and this finishes the proof. □

Remark. Combining 6.1 and 6.2 theorems we get that any integer programming problem defined by using the incidence matrix of a graph can be solved as a relaxed LP problem. This illustrates why these problems are easy to solve (in polynomial time), on the other hand the duality theorems can also be exploited. A detailed discussion on this is out of our scope. Instead, we show another application, namely finding **maximal matching** in bipartite graphs. We need the following theorem.

Theorem 6.3 *The incidence matrix of a bipartite graph is totally unimodular.*

Proof. Similar to the proof of theorem 6.2, we can use induction according to the size of subdeterminants □

The maximal matching problem can be formulated as the following integer programming problem

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ \sum_{e \sim u} x_e \quad & \leq 1 \quad (\forall u \in V) \\ x_e \quad & \in \quad \{0,1\} \end{aligned}$$

where $e \sim u$ denotes the incidence. Writing it a compact form and relaxing it we get the $\max \mathbf{1}^T x$, subject to $Ax \leq \mathbf{1}$, $x \geq 0$ problem, where A is the incidence matrix of the bipartite graph and $\mathbf{1}$ is the vector whose all element is 1. Since this is totally unimodular, the solution of the LP is integer.

6.4 Characterization of TU matrices

There do not seem to be any easily tested necessary and sufficient conditions for total unimodularity. There exist some characterization theorems for totally unimodular matrices. There is also an easily tested set of sufficient (but not necessary) conditions for total unimodularity. Here we provide two results, without discussing their proofs.

Theorem 6.4 (Camion, 1965) *A $0, \pm 1$ matrix A totally unimodular if and only if the sum of the elements of its each squared sub-matrix whose each row sum and column sum is even*[2] *is a multiple*

[2]Such matrices are called *Eulerian*.

of 4.

Theorem 6.5 (Ghouila-Houri, 1962) *An $m \times n$ integral matrix A is totally unimodular if and only if for each set $R \subseteq \{1, 2, \ldots, m\}$ can be divided into two disjoint sets R_1, R_2 such that*

$$\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{0, \pm 1\}, (j = 1, 2, \ldots, n).$$

Finally we provide a sufficient condition that can be easily checked.

Theorem 6.6 *A $0, \pm 1$ matrix A totally unimodular, if it contains no more than one $+1$ and no more than one -1 in each column.*

Remark. Theorem 6.2 immediately follows from above statement.

6.5 Exercises

6.5.1 Prove that if A matrix is totally unimodular, then multiplying any of its row or column by -1 the obtained matrix is still totally unimodular.

6.5.2 Prove that if A matrix is totally unimodular, then A^T is also totally unimodular.

6.5.3 The *edge-path incidence matrix*[3] of a graph is totally unimodular.

6.5.4 Let A be $0, \pm 1$ matrix, where in each column after the first cell that is 1 every cell is also 1 (top-down). Prove that A is totally unimodular.

6.5.5 Define a polyhedron ($\{x : Ax \leq b, x \geq 0\}$) with integer corner points, where A is an 3×3 matrix, the elements of A and b are integers, but A is not totally unimodular. Can we give an example, where A is a $0, \pm 1$ matrix. What is A is a $0, +1$ matrix?

[3]Each row of the matrix corresponds to an edge, while each column corresponds to a path of the graph. There is 1 in a cell, if the corresponding edge contained in the corresponding path, and 0 otherwise.

Chapter 7

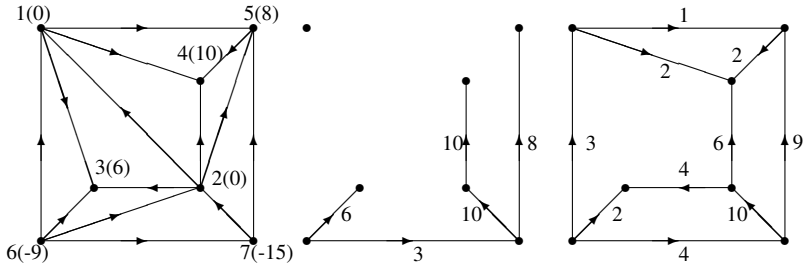
The transshipment problem

Given a directed graph G with $c : E(G) \to \mathbb{R}$ edge weighting. In this context $c(e)$ refers to the transportation cost on edge e per unit quantity. We assign numbers to the vertices of the graph too. If this number if positive we say the vertex is a target, if negative we say it is a source, and if zero we say that the vertex is an inner point. We may assume that sum of the number assigned to vertices is zero (if not, we can force this condition by adding a new vertex to the graph with an appropriate weight).

Suppose that we need to transport some cargo from source vertices to target vertices through to edges of the graph, such that the supply in each source and demand in each target is the absolute value of the assigned number (and zero in the inner points). The transportation cost in each edge is the cost of the edge multiplied by the transported quantity, while the total cost is the some of all transportation costs.

7.1 Problem formulation

Example. The numbers on the vertices are the names of them, the demand/supply marked in the brakets. In the two solutions we depicted only the edges that are involved in the transportation.



The transshipment problem can be written as an LP, where A is the **incidence matrix** of G, x_{ij} is the quantity transported on edge (i, j), c_{ij} transportation cost on edge (i, j) per unit quantity, while b vector represents the demand and supply of each vertex.

The LP formulation of the transshipment problem is the following.

$$\begin{array}{rrcl} & \min \sum\limits_{(i,j)\in E(G)} c_{ij}x_{ij} & = & c^T x \\ (*) & Ax & = & b \\ & x & \geq & 0 \end{array}$$

In our example matrix A is the following:

	1,3	1,4	1,5	2,1	2,3	2,4	2,5	5,4	6,1	6,2	6,3	6,7	7,2	7,5
1	-1	-1	-1	1	0	0	0	0	1	0	0	0	0	0
2	0	0	0	-1	-1	-1	-1	0	0	1	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0	1	0	0	0
4	0	1	0	0	0	1	0	1	0	0	0	0	0	1
5	0	0	1	0	0	0	1	-1	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	-1	-1	-1	-1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	-1	1

and

$$x_T = [x_{13}, x_{14}, x_{15}, x_{21}, x_{23}, x_{24}, x_{25}, x_{54}, x_{61}, x_{62}, x_{63}, x_{67}, x_{72}, x_{75}]$$

while

$$b_T = [0, 0, 6, 10, 8, -9, -15].$$

Observe that we can delete the last (or any) row of the matrix A and last coordinate of vector b in $(*)$ due to linear dependency.

The transshipment problem can be solved, for instance, in the following ways. If G is connected (aside from edge directions) then the edges, corresponding to the (positive) basic variables of any basic solution of $(*)$ LP, form a tree. Defining appropriate transformations of such trees leads to the **network simplex algorithm**. This algorithm is a variant of the well-known simplex algorithm that can be used effectively in practice. Other methods use the special structure of the problem more directly, at least theoretically, and exceed the performance of the simplex algorithm. We will provide more insights by analyzing a simpler task, the a **maximum flow problem**. Now we will prove the so called **integrality theorem**.

Theorem 7.1 (Integrality) *Assume that the each coordinate of the b vector of the above defined transshipment problem is integer. If the problem has feasible solution, then it has integer solution. If it has optimal solution, then it has integer optimal solution too.*

Proof. The fundamental theorem of LP says that if there is a feasible solution, then there is a basic feasible solution. Earlier we saw that A matrix is totally unimodular, therefore any feasible solution is integer. If there is an optimal solution, then there is an optimal basic solution, that is now necessarily integer. □

The transshipment problem and its formalism with the integrality theorem is a fortunate conjunction of operations research and combinatorics. In fact it is generalization of numerous earlier problem and model. Here we follow the reverse direction and discuss some problems as special cases of the general problem. First let us see a non-trivial application.

7.2 The caterer problem

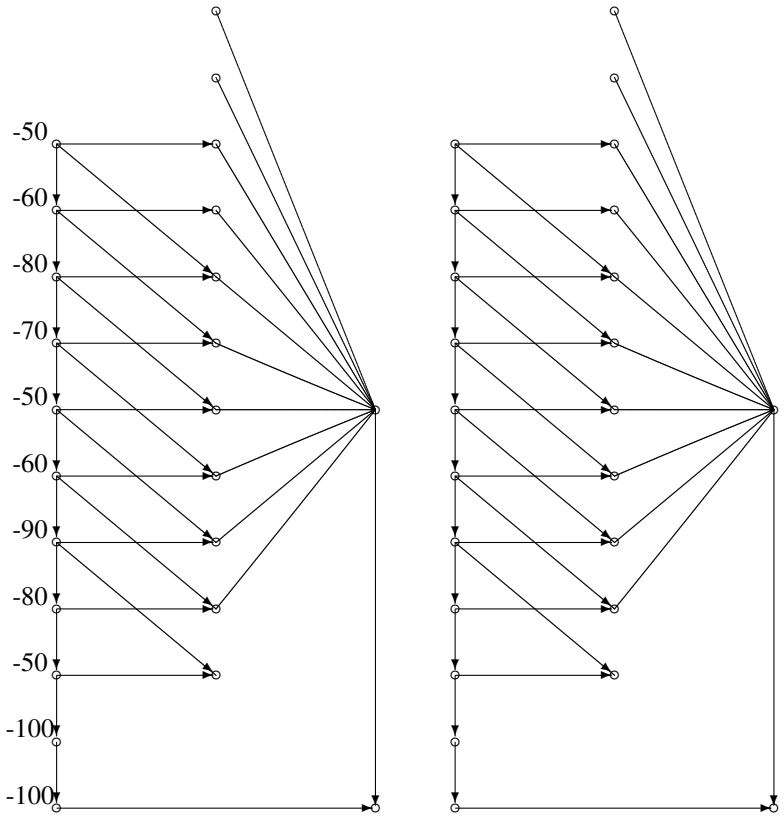
Originally the model is a *scheduling problem* developed to maintenance schedule of the US airforce's plains. Later, the inventors of the model (Jacobs, who published it as an engine restoration problem; Gaddum, Hoffman and Sokolowsky, who mentioned the problem as caterer problem; Prager, who investigated the case $q = p - 1$ as a transportation problem; and Ford and Fulkerson, who solved it as a network problem) published it as the *caterer problem* without mentioning its original application.

Given a canteen that needs clean napkins for n consecutive days. Each day j the number d_j of napkins needed is known in advance. The manager can buy new napkins (for a cent per piece) and re-use old ones (after washing them) to satisfy the demand. In case of washing he can choose from two options: the fast washing (q days, b cent per piece) and slow washing (p days, c cent per piece). Naturally, $p > q$ and $a > b > c$, the goal is to minimize the total cost.

Example: Let n=10, d_1=50, d_2=60, d_3=80, d_4=70, d_5=50, d_6=60, d_7=90, d_8=80, d_9=50, d_{10}=100; while p=4, q=2, a=200, b=75, c=25 be the parameters required in the caterer problem

Day	Re-used	Bought	Fast wash	Slow wash	Stored
1	0 (0)	50 (50)	50 (0)	0 (50)	0 (0)
2	0 (0)	60 (60)	60 (0)	0 (60)	0 (0)
3	50 (0)	30 (80)	50 (0)	30 (80)	0 (0)
4	60 (0)	10 (70)	60 (0)	0 (70)	10 (0)
5	50 (50)	0 (0)	60 (10)	0 (0)	0 (40)
6	60 (60)	0 (0)	60 (10)	0 (90)	0 (0)
7	90 (90)	0 (0)	50 (50)	0 (0)	40 (40)
8	60 (80)	20 (0)	100 (10)	0 (0)	20 (110)
9	50 (50)	0 (0)	0 (0)	0 (0)	70 (160)
10	100 (100)	0 (0)	0 (0)	0 (0)	170 (260)

The table shows a possible and an optimal (numbers in brackets) strategy. The optimal is an optimal basic solution of a transshipment problem assigned to caterer problem. We also show the graph that represents the task and its tree related to the optimal basic solution.



A transshipment model and tree of the optimal solution are the following. The vertices of the left side of the left figure represent the napkin used each day. From these vertices (as sources) napkins can be sent to re-use after 2 days by fast washing (horizontal edges, b is the cost), or re-use after 4 days by slow washing (skew edges, c is the cost), or can be stored until the next day (vertical edges, 0 cost). The d_j napkins required in day j can be supplied form node (storage) $j-q$ (fast washing), node $j-p$ (slow washing) or from the shop using an edge with cost a. It may be expedient to assume that there is enough number of napkins in the shop to cover the total demand. On the last day we put the remaining napkins (from the previous day and from the shop) to a virtual storage (with zero cost). By this description we formulate the caterer problem as a transshipment problem. On the right figure we show the edges correspond to the variables of the basis of the optimal basic solution. The numbers on the edges are the values of the variables. Observe the some variables are zero, means that the basic solution is degenerated. This phenomena frequently occurs in case of network problems. On the other hand, the integrality theorem guarantees that there is integer optimal solution, thus the obtained scheduling plan can be applied without numerical rounding.

7.3 Shortest path as transshipment

We have already discussed efficient algorithms to solve the shortest path problem. Now we just illustrate the power of the general transshipment model with the integrality theorem.

Assume that given a directed and edge weighted graph G and the task is to find the shortest $s-t$ path. Assign -1 demand to s and $+1$ demand to t. Consider that the edge weights are the costs solve the resulting transshipment problem, that is

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases} \qquad (i = 1, \ldots, n)$$

$$x_{ij} \in \{0, 1\} \qquad (1 \leq i, j \leq n).$$

If there is no negative cycle in G then the variables assigned to edges can only take the values 0 or 1. The edges for which the values of the corresponding variables takes the value 1 in the optimal solution designate the shortest path. (We note that partially the reverse is also true: finding shortest paths is an important sub-task of various algorithms for the transshipment problem.)

7.4 Transportation problem

The problem was first investigated by F. I. Hitchcock in 1941, and sometimes it is called **Hitchcock's transportation problem**:

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = r_i, \quad (i = 1, 2, \ldots, m)$$

$$\sum_{i=1}^{m} x_{ij} = s_j, \quad (j = 1, 2, \ldots, n)$$

$$x_{ij} \geq 0$$

It is easy to see, that it is a transshipment problem defined on a **bipartite graph**. Let us define G such that $V(G) = U \cap V$, where $U = \{u_1, u_2, \ldots, u_m\}$ are the sources, $V = \{v_1, v_2, \ldots, v_m\}$ is the set of target nodes, while $E(G) = \{u_i v_j : u_i \in U, v_j \in V\}$. Furthermore let the supply be r_i at node u_i ($i = 1, \ldots, m$), and the demand be s_j at node v_j ($j = 1, \ldots, n$). The transportation problem can be solved according to the integrality theorem. Besides the most straightforward one the transportation problem has many other applications. There discussion is beyond the scope of this lecture.

7.5 Assignment problem

We introduce the problem through an example. Suppose that there are 5 men and 5 different task, and the performance (ability) of man i on the task j can be given by single number ($0 \leq i, j \leq 5$). The problem is to assign each man to exactly one task such that the sum of performances is to be maximized. The man-task performance matrix is given as

	1	2	3	4	5
1	7	5	4	4	5
2	7	9	7	9	4
3	4	6	5	8	5
4	5	4	5	7	4
5	4	5	5	8	9

We can model the problem as an integer programming task. Let $x_{ij} = 1$, if ith man gets the task j and $x_{ij} = 0$ otherwise. The constraints are $\sum_{j=1}^{5} x_{ij} = 1$ $(i = 1, \ldots, 5)$ (each men gets exactly one task) and $\sum_{i=1}^{5} x_{ij} = 1$ (j=1, ..., 5) (each task is done by exactly one men). The objective function is $\sum_i \sum_j c_{ij} x_{ij}$. In general, the assignment problem is defined by the IP

$$\max \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$(*) \qquad \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, n)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, \ldots, n)$$

$$x_{ij} \in \{0, 1\} \quad (1 \leq i, j \leq n).$$

Observe that $(*)$ has *always* $n!$ feasible solutions and one of those of course optimal. Due to the integrality theorem the optimal basic solution of the following transshipment problem is the optimal solution of $(*)$ too:

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} (-c_{ij}) x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

$$x_{ij} \leq 0 \qquad (1 \leq i, j \leq n).$$

We should mention here that P. R. Halmos and H. Vaughon in 1950 published a quite absurd application of $(*)$ problem. There, the marriage of n men and n women need to be optimized according to a wistfulness point c_{ij} assigned to each man i by each woman j (where n is the best score, 1 is the worst, and a woman gives different points to each man). The total happiness is measured via $\sum_i \sum_j c_{ij}$-vel. The problem and the corresponding integrality theorem can be applied well to *matching problems of bipartite graphs*.

Recall that graph G is **bipartite**, if $V(G) = A \cup B$, where $A \cap B = \emptyset$, and every edge connects a vertex from A and a vertex from B. A **matching** of a graph is a set of edges without common vertices. It is easy to see that a maximal M matching (i.e. the maximal cardinality) of a bipartite G graph can be obtained by solving the following IP:

$$\begin{aligned}
\max \qquad & \sum_{j \in B} \sum_{i \in A} x_{ij} \\
& \sum_{j:(i,j) \in E(G)} x_{ij} \leq 1 \quad (i \in A) \\
& \sum_{i:(i,j) \in E(G)} x_{ij} \leq 1 \quad (j \in B) \\
& x_{ij} \leq 0 \qquad (i \in A, j \in B),
\end{aligned}$$

where M contains such (i, j) edges, where $x^*_{ij} = 1$ in the optimal solution. **Dénes König's famous theorem** can also be readily obtained in a

Theorem 7.2 *Let G be a bipartite graph where $|A| = |B| = n$ and $d(v) = k \geq 1$ for each $v \in V(G)$. Then G has a perfect matching (i.e. a matching that matches all vertices of the graph).*

Proof. Define a transshipment problem to the graph G as follows. Assign -1 to each point of A and $+1$ to each point of B. Let the direction of edges be from $A$to B (and now there is no cost on the edges). The obtained problem *has* a feasible solution x: simply let $x_e = 1/k$ for each edge e. Thus, due to the integrality theorem there is an x^* integer feasible solution, where $x^*_e \in \{0, 1\}$ for each edge e. In fact, there is exactly one edge from each node in A, where the corresponding variable is 1. These edges are the elements of M perfect matching. □

7.6 Summary

In this section we give a brief summary of the LP models of the most important **network problems**.

Network $G = (V, E)$ has **nodes** V and **edges** E. Each edge (i, j) has a capacity u_{ij} and a cost c_{ij}. Each vertex i provides a net supply b_i.

An $X, \overline{X}$ cut is a partition of $V(G)$ such that any node is either $s \in X$ or $t \in \overline{X} = V(G) \backslash X$. The set of edges of the cut is denoted by $E(X, \overline{X})$. The **capacity** of cut $X, \overline{X}$ is $c(X, \overline{X}) := \sum_{i \in X, j \in \overline{X}} c_{ij}$. Where applicable, there are two distinguished nodes: $s =$ **source** and $t =$ **sink**.

7.6.1 Minimum spanning tree

Primal.

$$\begin{aligned} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ & \sum_{(i,j) \in E(X, \overline{X})}^{n} x_{ij} \quad > 0 \qquad (\forall X \subset V, X \neq \emptyset) \\ & x_{ij} \quad \in \{0, 1\} \quad (1 \leq i, j \leq n). \end{aligned}$$

Feasibility: if there is no $X \subseteq V$ ($\emptyset \neq X \neq V$) such that $E(X, \overline{X}) = \emptyset$.

7.6.2 Shortest path problem

Primal.

$$\begin{aligned} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ & \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} \quad = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases} \qquad (i = 1, \ldots, n) \\ & x_{ij} \quad \in \{0, 1\} \qquad (1 \leq i, j \leq n). \end{aligned}$$

Dual.

$$\begin{aligned} \max & \, y_s - y_t \\ & y_i - y_j \quad \leq c_{ij} \qquad \forall (i, j) \in E \\ & y_i \quad \text{unrestricted} \quad \forall i \in V \end{aligned}$$

Feasibility: if there is no $X \subseteq V$ with $s \in X$, $t \in \overline{X}$ such that $E(X, \overline{X}) = \emptyset$.

7.6.3 Maximum-flow problem

Primal.

$$\begin{aligned} & \max z \\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} &= \begin{cases} z, \text{ if } i = s \\ -z, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases} && (i = 1, \ldots, n) \\ 0 \leq x_{ij} &\leq u_{ij} && (1 \leq i, j \leq n). \end{aligned}$$

Dual.

$$\begin{aligned} \min \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} v_{ij} & \\ y_i - y_j + v_{ij} &\geq 0 && \forall (i,j) \in E \\ y_t - y_s &= 1 && \\ v_{ij} &\geq 0 && \forall (i,j) \in E \\ y_i &\quad \text{unrestricted} && \forall i \in V \end{aligned}$$

Feasibility: no flow bigger than the capacity of the corresponding cut.

7.6.4 Minimum-cost (s, t)-flow problem

Primal.

$$\begin{aligned} & \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} &= \begin{cases} f, \text{ if } i = s \\ -f, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases} && (i = 1, \ldots, n) \\ 0 \leq x_{ij} &\leq u_{ij} && \forall (i,j) \in E \end{aligned}$$

Dual.

$$\begin{aligned} \max f y_s - f y_t - \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} v_{ij} & \\ y_i - y_j - v_{ij} &\leq c_{ij} && \forall (i,j) \in E \\ v_{ij} &\geq 0 && \forall (i,j) \in E \\ y_i &\quad \text{unrestricted} && \forall i \in V \end{aligned}$$

Feasibility: there is no set $X \subseteq V$ with $s \in X$ and $t \in \overline{X}$ such that $f > \sum_{(i,j) \in E(X, \overline{X})} u_{ij}$.

7.6.5 Transshipment problem

Primal.

$$\begin{aligned} \max & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} & = b_i \quad (i = 1, \ldots, n) \\ x_{ij} & \geq 0 \quad \forall (i,j) \in E \end{aligned}$$

Dual.

$$\begin{aligned} \min & \sum_{i \in V} b_i y_i \\ y_i - y_j & \leq c_{ij} \qquad \forall (i,j) \in E \\ y_i & \text{ unrestricted} \quad \forall i \in V \end{aligned}$$

Feasibility: there is no set $X \subseteq V$ such that $\sum_{i \in X} b_i > 0$ and $E(X, \overline{X}) = \emptyset$

7.6.6 Minimum cost network flow problem

Primal.

$$\begin{aligned} \min & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} & = b_i \quad (i = 1, \ldots, n) \\ 0 \leq x_{ij} & \leq u_{ij} \quad \forall (i,j) \in E \end{aligned}$$

Dual.

$$\begin{aligned} \max & \sum_{i \in V} b_i y_i - \sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij}v_{ij} \\ y_i - y_j - v_{ij} & \leq c_{ij} \qquad \forall (i,j) \in E \\ v_{ij} & \geq 0 \qquad \forall (i,j) \in E \\ y_i & \text{ unrestricted} \quad \forall i \in V \end{aligned}$$

Feasibility: there is no set $X \subseteq V$ such that $\sum_{i \in X} b_i > \sum_{(i,j) \in E(X, \overline{X})} u_{ij}$.

7.7 Exercises

7.7.1 Give an example of a minimum-cost network flow problem with all arc costs positive and the following counter intuitive property: if the supply at a particular source node and the demand at a particular sink node are simultaneously reduced by one unit, then the optimal cost increases.

7.7.2 (Vanderbei's book, Ch.14. Exercises) Bob, Carol, David, and Alice are stranded on a desert island. Bob and David each would like to give their affection to Carol or to Alice. Food is the currency of trade for this starving foursome. Bob is willing to pay Carol 7 clams if she will accept his affection. David is even more keen and is willing to give Carol 9 clams if she will accept it. Both Bob and David prefer Carol to Alice (sorry Alice). To quantify this preference, David is willing to pay Alice only 2 clams for his affection. Bob is even more averse: he says that Alice would have to pay him for it. In fact, she'd have to pay him 3 clams for his affection. Carol and Alice, being proper young women, will accept affection from one and only one of the two guys. Between the two of them they have decided to share the clams equally between them and hence their objective is simply to maximize the total number of clams they will receive. Formulate this problem as a transportation problem. Solve it.

7.7.3 (Stacho' lecture notes, Sec. 10.1)A new car costs \$12,000. Annual maintenance costs are as follows: $m_1 = \$2,000$ first year, $m_2 = \$4,000$ second year, $m_3 = \$5,000$ third year, $m_4 = \$9,000$ fourth year, and $m_5 = \$12,000$ fifth year and on. The car can be sold for $s_1 = \$7,000$ in the first year, for $s_2 = \$6,000$ in the second year, for $s_3 = \$2,000$ in the third year, and for $s_4 = \$1,000$ in the fourth year of ownership. An existing car can be sold at any time and another new car purchased at \$12,000. What buying/selling strategy for the next 5 years minimizes the total cost of ownership?

7.7.4 (Winston's book, Sec. 7.1.) Powerco has three electric power plants that supply the needs of four cities. Each power plant can supply the following numbers of kilowatt-hours (kwh) of electricity: plant 1-35 million; plant 2-50 million; plant 3-40 million (see table). The peak power demands in these cities, which occur at the same time (2pm), are as follows (in kwh): city 1-45 million; city 2-20 million; city 3-30 million; city 4-30 million. The costs of sending 1 million kwh of electricity from plant to city depend on the distance the electricity must travel. Formulate an LP to minimize the cost of meeting each city's peak power demand.

Machine/Time	Job 1	Job 2	Job 3	Job 4
1	14	5	8	7
2	2	12	6	5
3	3	8	3	9
4	2	4	6	10

7.7.5 (Winston's book, Sec. 7.5) Machineco has four machines and four jobs to be completed. Each machine must be assigned to complete one job. The time required to set up each machine for completing each job is shown in table. Machineco wants to minimize the total setup time needed to complete the four jobs. Use linear programming to solve this problem.

From/To	City 1	City 2	City 3	City 4	Supply
Plant 1	8 ($)	6	10	9	35
Plant 2	9	12	13	7	50
Plant 3	14	9	16	5	40
Demand (m kwh)	45	20	30	30	

7.7.6 (Winston's book, Sec. 7.6 - Problems) General Ford produces cars at L.A. and Detroit and has a warehouse in Atlanta; the company supplies cars to customers in Houston and Tampa. The cost of shipping a car between points is given in the table 60m where "-" means that a shipment is not allowed). L.A. can produce as many as 1,100 cars, and Detroit can produce as many as 2,900 cars. Houston must receive 2,400 cars, and Tampa must receive 1,500 cars.

1. Formulate a transportation problem that can be used to minimize the shipping costs incurred in meeting demands at Houston and Tampa.
2. Modify the model if shipments between L.A. and Detroit are not allowed.
3. Modify the model if shipments between Houston and Tampa are allowed at a cost of $5.

From/To	L.A	Detroit	Atlanta	Houston	Tampa
L.A	0	140	100	90	225
Detroit	145	0	111	110	119
Atlanta	105	115	0	113	78
Houston	89	109	121	0	–
Tampa	210	117	82	–	0

Chapter 8

Stochastic problems

Stochastic optimization problems are optimization problems that involve uncertainty. Whereas deterministic optimization problems are formulated with known parameters, "real world" problems often include some unknown parameters. Stochastic programming models are similar in style to linear (or non-linear) programming models, but take advantage of the fact that probability distributions governing the data are known or can be estimated. In this chapter give a brief introduction the field by discussing some classical stochastic problems and some of their possible mathematical models.

8.1 Newsboy problem

A newsboy sells newspapers at the corner of Dóm square Szeged, and each day she must determine how many newspapers to order. She pays the company 20 (cent) for each paper and sells the papers for 25 each. Newspapers that are unsold at the end of the day are worthless. She knows that each day she can sell between six and ten papers, with each possibility being equally likely.

Note that two different problem may occur:

1. Suppose that she order 8 but sells just 6 by the end of the day. Then she pays $8 \times 20 = 160$ but her income is only $6 \times 25 = 150$, thus she will be in negative 10.

2. If she orders 6, but there would be 8 buyer, then her profit will be $6 \times (25 - 20) = 30$, however it could have been more, and she may loose prospective customers whom she cannot serves.

	Papers demanded				
Papers ordered	6	7	8	9	10
6	30	30	30	30	30
7	10	35	35	35	35
8	-10	15	40	40	40
9	-30	-5	20	45	45
10	-50	-25	0	25	50

8.1.1 Solution

We present four decision criteria that can be used here to **make decision under uncertainty.**

The set $S = \{6, 7, 8, 9, 10\}$ contains the possible values of the daily demand for newspapers. We are given that $p_6 = p_7 = p_8 = p_9 = p_10 = 1/5$, that is the probability of a certain number of buyers come (note that the probabilities can be given by any probability distribution).

The vendor should determine the supply, that can be $6, 7, 8, 9$ or 10 (obviously it is not worth to order less than 6 or more than 10 for her!).

If she purchases i papers and j papers are demanded, then i papers are purchased at a cost of $20i$ cent, and $\min(i, j)$ papers are sold for 25 cent each. Then the profit is

$$r_{ij} = \begin{cases} 25i - 20i = 5i, & \text{if} \quad i \leq j \\ 25j - 20i, & \text{if} \quad i \geq j \end{cases}$$

The following table shows the possible **payoffs** (profits).

Maximin criterion

According to this criterion, the vendor orders i papers such that $\min_{j \in S} r_{ij}$ is maximal. For each action, determine the worst outcome (smallest reward). The **maximin criterion** chooses the action with the "best" worst outcome.

Thus, the maximin criterion recommends ordering 6 papers. This ensures that she will, no matter what will be the daily demand, earn a profit of at least 30 cent.

Maximax criterion

According to this criterion, the vendor orders i papers such that $\max_{j \in S} r_{ij}$ is maximal. This provides the maximum potential payoff (with high risk, naturally). For each action, determine the

Papers ordered	Worst case	Reward
6	6,7,8,9,10	30
7	6	10
8	6	-10
9	6	-30
10	6	-50

best outcome (largest reward). The **maximax criterion** chooses the action with the "best" best outcome.

Papers ordered	Best case	Best outcome
6	6,7,8,9,10	30
7	7,8,9,10	35
8	8,9,10	40
9	9,10	45
10	10	50

Thus, the maximax criterion would recommend ordering 10 papers. In the best case (when 10 papers are demanded), this yields a profit of 50 cent. Of course, making a decision according to the maximax criterion leaves vendor open to the worst possibility that only 6 papers will be demanded, in which case she loses 50 cent.

Minimax regret

The **minimax regret criterion** (developed by L. J. Savage) uses the concept of opportunity cost to arrive at a decision. For each possible demand j it first determines the best action, i.e. the action maximizes r_{i^*j}. Then the opportunity loss (or regret) is $r_{i^*j} - r_{ij}$ for each possible action i.

For example, if $j = 7$ papers are demanded, the best decision is to order 7 papers, yielding a profit of $r_{77} = 35$. Suppose the vendor chooses to order $i = 6$ papers instead of 7. Since $r_{67} = 30$, the opportunity loss, or regret for $i = 6$ and $j = 7$ is 35-30= 5. The regret table is given as follows:

The minimax regret criterion chooses an action by applying the minimax criterion to the regret matrix. In other words, the minimax regret criterion attempts to avoid disappointment over what might have been. The minimax regret criterion recommends ordering 6 or 7 papers.

	Demand				
Papers ordered	6	7	8	9	10
6	$30-30=0$	$35-30=5$	$40-30=10$	$45-30=15$	$50-30=\mathbf{20}$
7	$30-10=\mathbf{20}$	$35-35=0$	$40-35=10$	$45-35=10$	$50-35=15$
8	$30+10=\mathbf{40}$	$35-15=20$	$40-40=0$	$45-40=5$	$50-40=10$
9	$30+30=\mathbf{60}$	$35+5=40$	$40-20=20$	$45-45=0$	$50-45=5$
10	30 + 50 = **80**	$35+25=60$	$40-0=40$	$45-25=20$	$50-50=0$

Expected value criterion

The **expected value criterion** chooses the action that yields the largest expected reward. Suppose that the demand is given by the probability distribution $(p_6, p_7, \ldots, p_10)$, where $p_i \geq 0$ is the probability that the demand will be i ($i = 6, \ldots, 10$) and $\sum_{i=6}^{10} p_i = 1$. For instance, in case of $p_i = 1/5$ ($i = 6, \ldots, 10$), the expected outcomes are:

Papers ordered	Expected outcome.
6	$\frac{1}{5}(30+30+30+30+30)=30$
7	$\frac{1}{5}(10+35+35+35+35)=30$
8	$\frac{1}{5}(-10+15+40+40+40)=25$
9	$\frac{1}{5}(-30-5+20+45+45)=25$
10	$\frac{1}{5}(-50-25+0+25+50)=0$

and this would recommend ordering 6 or 7 papers. In the general case the expected values $\sum_i p_i r_{ij}$ values should be calculated for each j.

8.1.2 General problem of discrete demand

Let c be the **vendor price**, d be the **selling price** ($d > c$), the **demand** is given by $(p_1, \ldots, p_k)$ probability distribution, where

$$p_i = \Pr(x_i \text{ the demand}).$$

Let X be a random variable such that $\Pr(X = x_i) = p_i$ (thus X represents the demand). Then the **expected profit** is

$$\mathbb{E}[(d-c)X].$$

Suppose that the vendor orders t papers. The goal is to determine the value t such that the expected profit would be maximal.

- if $t \geq x \in (x_1, \ldots, x_n)$ then $dx - ct = -c(x-t) + (d-c)x$
- if $t < x \in (x_1, \ldots, x_n)$ then $dt - ct = -(d-c)(x-t) + (d-c)x$

We can see that maximizing the profit is equivalent to the minimization of the following function:

$$g(t) = \sum_{x_i : t > x_i} c(t - x_i)p_i + \sum_{x_i : t \leq x_i} (d-c)(x_i - t)p_i$$

The first term is the loss due to the **papers not sold** while the second term represents the **opportunity cost**. If X would follow not a discrete but a continuous probability distribution then, similarly as in the discrete case, we obtain

$$g(t) = c \int_{-\infty}^{t} (t-x)f(x)dx + (d-c) \int_{t}^{\infty} (x-t)f(x)dx$$

and the goal is the minimization of $g(t)$. From this, we can get

$$g(t) = c \int_{-\infty}^{t} (t-x)f(x)dx + c \int_{t}^{\infty} (t-x)f(x)dx + d \int_{t}^{\infty} (x-t)f(x)dx =$$

$$= ct \int_{-\infty}^{\infty} f(x)dx - c \int_{-\infty}^{\infty} f(x)dx - dt \int_{t}^{\infty} f(x)dx + d \int_{t}^{\infty} xf(x)dx.$$

Differentiating according to t we get

$$g'(t) = c - d \int_{t}^{\infty} f(x)dx + dtf(t) - dtf(t) = d(F_X(t) - 1) + c,$$

where $F_X(t) = \int_{-\infty}^{t} f(x)dx$ is the probability distribution function of X. The minimum of g is obtained by solving the $g'(t) = 0$, that is $t = F^{-1}((d-c)/d)$.

An LP model

Let d is the selling price and c is the vendor price ($d > c$) per unit as before, X is the random variable stands for the demand, and the vendor orders quantity t of a certain product (e.g. a newspaper).

If $X = x > t$, the a "back order penalty" $b \geq 0$ per unit is incurred. The total cost of this is equal to $b(x-t)$ if $x > t$ and zero otherwise. Similarly, if $X = x < t$, then a "holding cost" h is incurred. The total cost of this is equal to $h(t-x)$ if $x < t$ and zero otherwise. The cost function is then

$$g(t,x) = ct + b[x-t]_+ + h[t-x]_+,$$

where $[a]_+ = \max\{a, 0\}$. The objective is to minimize $g(t,x)$ First, let $X \equiv x$ a constant parameter. The objective function can be rewritten as

$$g(t,x) = \max\{(c-b)t + bx, (c+h)t - hx\}$$

which is a piecewise linear function with minimum attained at $t = x$. Evidently, if the demand x is known in advance, the best decision is to order exactly the demand quantity $t = x$. We can formulate the problem as an LP:

$$\begin{array}{lrcl} \min z = & y & & \\ \text{st} & y & \geq & (c-b)t + bx \\ & y & \geq & (c+h)t - hx \\ & x & \geq & 0 \end{array}$$

Next, we consider now the case when the ordering decision should be made before a realization of the demand becomes known. Thus, the demand X is now a random variable. In this case, it is usually assumed that the probability distribution of X is known (e.g. estimated statistically from historical data). The corresponding optimization problem is

$$\min_t \mathbb{E}[g(t,X)]$$

that is, minimizing the total cost "on average". If the distribution of X is discrete, and it takes values $x_1, \ldots, x_k$ with probabilities $p_1, \ldots, p_k$ ($p_i \geq 0$, $i = 1, \ldots, k$ and $\sum_{i=1}^k p_i = 1$), then the expected value is

$$\mathbb{E}[g(t,X)] = \sum_{i=1}^k p_i g(t, x_i).$$

The problem can be formulated as an LP as follows:

$$\begin{array}{lrcll} \min z = & \sum_{i=1}^k p_i y_i & & & \\ \text{st} & y_i & \geq & (c-b)t + bx_i & i = 1, \ldots, k \\ & y_i & \geq & (c+h)t - hx_i & i = 1, \ldots, k \\ & x_i & \geq & 0 & i = 1, \ldots, k \end{array}$$

8.2 Reliability

8.2.1 Network reliability

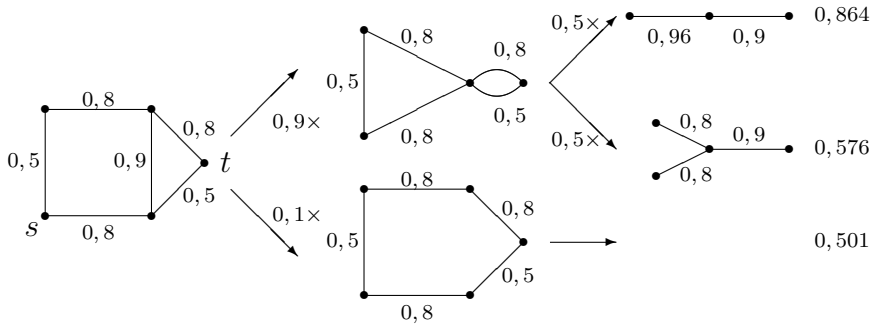
Similarly to the largest probability path problem, we are given a graph G with two distinguished nodes s and t, and each edge e is assigned a probability p_e. In our interpretation this is the probability of the usability or existence the edge. Existence of an edge is independent to the existence of other edges. We saw that the finding the largest probability $s-t$ is equivalent to finding the shortest $s-t$ path after transforming the weights appropriately. However, it is much more difficult to calculate the reaching probability of t starting from s: there is not known any efficient algorithm to solve the problem, moreover, it is not hoped to find one soon. Fortunately, for not too big graphs the problem can be handled using a recursion algorithm (that requires exponential running time in the number of edges.)

Let G be the given graph and chose an edge e exists with probability p_e. Let $G \setminus e$ be the graph obtained from G by deleting e and G/e be the graph obtained from G by contracting e. Then, since

$$\Pr(G) = (1-p_e)Pr(G \setminus e) + p_e \Pr(G/e),$$

the original problem is reduced to two smaller problems.

Example.



Observe that a parallel edge pair, with assigned probabilities p_1 and p_2, can be substituted by one edge with probability $p_1 + p_2 - p_1 p_2$. The calculation for the lower branch goes similarly. Moreover, if we arrive to an already calculated subgraph we can simply use the result obtained previously. Thus $\Pr(G) = 0.72 \times 0.9 + 0.501 \times 0.1 = 0.6981$.

8.2.2 System reliability

Reliability of a system or a product is defined as the probability that the product will not fail throughout a prescribed operating period. Most products are made up of a number of components. The reliability of each component and the configuration of the system consisting of these components determines the system reliability (i.e., the reliability of the product).

Suppose now, that a system (product) has n different components and all should work together properly. The components go to ruin independently. For a component i, we have two options:

1. cheaper: price s_i with probability of proper working q_i;
2. more expensive: price r_i with probability of proper working p_i.

Assume that $s_i < r_i$ and $q_i < p_i$ $(i = 1, \ldots, n)$. According to technical obligations the system should work with probability p. The goal is to choose the components properly with minimum cost.

We can formulate the problem as a $0-1$ IP. Let $x_i \in \{0, 1\}$ such that $x_i = 0$ if we choose the component which price is r_i and $x_i = 1$ if we choose the component which price is s_i. Then, the probability that the system works properly is

$$\prod_{i=1}^{n} q_i \prod_{i=1}^{n} \left(\frac{p_i}{q_i}\right)^{x_i},$$

while the cost is

$$\sum_{i=1}^{n} (r_i - s_i)x_i + \sum_{i=1}^{n} s_i.$$

By introducing $b = \log(p/\prod_{i=1}^{n} q_i)$ and $a_i \log(p_i/q_i)$ we can formulate the following LP for the problem:

$$\begin{array}{lrcl} \min z = & \sum_{i=1}^{n} (r_i - s_i)x_i & & \\ \text{st} & \sum_{i=1}^{n} a_i x_i & \geq & b \\ & x_i & \in & \{0, 1\} \quad i = 1, \ldots, n \end{array}$$

That is a knapsack problem and it can be solved using e.g. the branch-and-bound method.

8.3 Portfolio Selection problem

Suppose that we are planning to invest our capital to various assets (i.e. activities, bonds, stocks, or properties, etc). Consider n different assets where asset i will give a return R_i (that is a random

variable) and relative amount of capital invested in i is x_i ($i = 1 \ldots, n$). Let $x = (x_1, x_2, \ldots, x_n)$ be the portfolio vector with $\sum_{i=1}^{n} x_i = 1$. The **expected return of the portfolio** is calculated as

$$R_p = \sum_{i=1}^{n} \mathbb{E}[R_i] x_i = \mu_i x_i.$$

Maximizing the expected return of a portfolio can be formulated as a knapsack problem:

$$\begin{array}{lrcl} \max R_p = & \sum_{i=1}^{n} \mu_i x_i & & \\ \text{st} & \sum_{i=1}^{n} x_i & = & 1 \\ & x_i & \geq & 0 \quad i = 1, \ldots, n \end{array}$$

If $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$, then the optimal solution of the problem is $x_1 = 1, x_2 = x_3 = \cdots = x_n = 0$. In reality, following this strategy leads to a bankruptcy with probability 1.

8.3.1 The Markowitz model

Markowitz portfolio theory provides a method to analyze how good a given portfolio is based on only the means and the variance of the returns of the assets contained in the portfolio. An investor is supposed to be **risk-averse**, hence he wants a small variance of the return (i.e. a small risk) and a high expected return. The **variance of the portfolio** is calculated as

$$\sigma_p^2 = \mathrm{Var}[\sum_{i=1}^{n} R_i x_i] = \mathbb{E}[(\sum_{i=1}^{n} (R_i - \mu_i) x_i)^2] = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j,$$

respectively, where σ_{ij} is the covariance between asset i and j (while σ_{ii} is the variance of asset i).

A portfolio is called **efficient** (or optimal) if it provides the best expected return on a given level of risk, or alternatively, the minimum risk for a given expected return. The Markowitz model (1952) is a **quadratic programming problem** with the goal of minimizing the variance (risk) given a desired value $\rho > 1$ of return:

$$\begin{array}{lrcl} \min \quad \sigma_p = & \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j & & \\ \text{st} & R_p = \sum_{i=1}^{n} \mu_i x_i & \geq & \rho \\ & \sum_{i=1}^{n} x_i & = & 1 \\ & x_i & \geq & 0 \quad i = 1, \ldots, n \end{array}$$

Remark. Observe that the objective function of the model is a quadratic function of the decision variables. We do not deal with the solution of such **non-linear problem**, just mention that there are many efficient algorithms available to solve such problems. Another difficulty is to estimate the individual asset returns and variances and covariances.

Estimating the expected return and variance

A classical approach is to consider the price time series of n assets, where the closure price of asset i at time t is denoted by $P_i(t)$ $(t = 0, 1, \ldots, T)$. The **daily logarithmic return** of i is defined as

$$R_i(t) = \log \frac{P_i(t)}{P_i(t-1)} = \log P_i(t) - \log P_i(t-1), (t = 1, \ldots, T)$$

Often, the expected return of asset i is estimated as the average of the past daily returns, thus

$$\overline{\mu}_i = \frac{1}{T} \sum_{t=1}^{T} R_i(t).$$

In case of stationary independent normally distributed returns this is the maximum likelihood estimator of the return.

Remark. We note that there are other estimators, for instance the James-Stein estimator.

The covariance between asset i and j can be estimated by the formula

$$\hat{\sigma}_{i,j}^2 = \frac{1}{T} \sum_{t=1}^{T} (R_i(t) - \overline{\mu}_i)(R_j(t) - \overline{\mu}_j).$$

8.3.2 The Mean Absolute Deviation model

We may avoid the difficulty of both the non-linear objective function and estimating the elements of the covariance matrix $\Sigma = (\sigma_{ij})_{i,j}$ if instead minimizing the covariance we minimize the **mean absolute deviation**

$$\mathbb{E}[|\sum_{i=1}^{n} (R_i - \mu_i)x_i|].$$

Konno and Yamazaki provided a method (in 1990) with this goal by avoiding the calculation of μ_i and, even more importantly, σ_{ij} values. Let

$$\overline{R}_i = \frac{1}{T} \sum_{t=1}^{T} R_i(T) \text{ and } a_i(t) = R_i(t) - \overline{R}_i.$$

The portfolio optimization problem can be written in the form

$$\begin{array}{lrcll} \min & \frac{1}{T} \sum_{t=1}^{T} |\sum_{i=1}^{n} a_i(t)x_i| & & & \\ \text{st} & \sum_{i=1}^{n} \overline{R}_i x_i & > & \rho & \\ & \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i & \geq & 0 & i = 1, \ldots, n \end{array}$$

This is not an LP (because of the absolute value in the objective function), but it can be rephrased to an LP using simple tricks:

$$\begin{array}{lrcll} \min & \frac{1}{T}\sum_{t=1}^{T} y_t & & & \\ \text{st} & -y_t \leq \sum_{i=1}^{n} a_i(t)x_i & \leq & y_t & t = 1, \ldots, T \\ & \sum_{i=1}^{n} \overline{R}_i x_i & > & \rho & \\ & \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i & \geq & 0 & i = 1, \ldots, n \end{array}$$

8.4 Exercises

8.4.1 Solve the news vendor problem for the following scenario. The vendor price of the newspaper is $c = 50$, the selling price is $d = 70$. According to past observations the demand X is between 40 and 60 uniformly at random.

8.4.2 (Winston's book Ch. 13 problems) Pizza King and Noble Greek are two competing restaurants. Each must determine simultaneously whether to undertake small, medium, or large advertising campaigns. Pizza King believes that it is equally likely that Noble Greek will undertake a small, a medium, or a large advertising campaign. Given the actions chosen by each restaurant, Pizza King's profits are as shown in the following table. For the maximin, maximax, and minimax regret criteria, determine Pizza King's choice of advertising campaign.

	Noble Greek chooses		
Pizza King chooses	small	medium	large
small	6000 ($)	5000	2000
medium	5000	6000	1000
large	9000	6000	0

8.4.3 Assume that the following two assets yield to following yearly returns How to divide and

	Year 1	Year 2	Year 3
Property	0.05	-0.03	0.04
Security	-0.05	0.21	-0.10

invest our money to the different assets according to the Markowitz model (using the standard statistical estimators of return and variance)?

8.4.4 Solve the Markowitz portfolio optimization problem for two variables analytically using the *Lagrange-method*.

8.4.5 Solve the Markowitz portfolio optimization problem in the general case using the *Lagrange-method*.

Chapter 9

Game theory

Game Theory is the study of decision making under competition. More specifically, Game Theory is the study of optimal decision making under competition when one individual's (agent's) decisions affect the outcome of a situation for all other individuals involved. Here we only deal with Classical Game Theory, that focuses on optimal play in situations where one or more people must make a decision and the impact of that decision and the decisions of those involved is known. It has a wide-range applications in many fields of social science and economics, as well as in logic and computer science.

In this chapter we discuss the main definitions and concepts of the topic, zero-sum games and their relationship with linear programming. Finally we give a short introduction to non-zero sum games too.

9.1 Pure and mixed strategies

9.1.1 Pure strategies

Two TV networks compete for an audience of 100 million viewers in a specific time slot. The networks announce their schedule ahead of time and do not know of the decision of the other until the show time. Based on that a certain number of people will tune to N1 while the rest will watch N2. The market research revealed the following expected number of viewers of N1.

For example, if N1 shows Western while N2 shows a Comedy, then 60 million people will watch N1, and $100 - 60 = 40$ will watch N2. The question is that what strategy should the net-

	N2		
N1	Western	Action	Comedy
Western	35	15	60
Action	45	58	50
Comedy	38	14	70

works use to maximize their viewership?

The following terminology is used.

- N1 is called **row player**
- N2 is called **column player**
- The above defined matrix is called **payoff matrix** (of the row player)
- $\Sigma_1 = \{$Western, Action, Comedy$\}$ $(= \Sigma_2)$ is the set of strategies of the row player (and column player, resp.)
- The game is **constant-sum game** if the outcome for both players sums up to a constant (now 100 million, but usually 0) in each case (i.e. strategy pairs)

Let us look at the structure of **outcomes**.

- For instance, if N1 chooses to show Western, then it can expect 60 million viewers if N2 chooses to show a Comedy; however it can expect only 15 million if N2 shows an Action $\Rightarrow$ this choice can guarantee at most 15 million for N1 in the **worst case**.
- If the N1 instead chooses to show a Comedy, the situation is even worse, since then only 14 million viewers are guaranteed by expectations (that is the minimum in the 3rd row).
- Since N1 does not know what N2 will show, the best is to choose to show an Action in which case 45 million or more viewers will tune to N1 (regardless of what N2 does.)

Observe that in this strategy N1 (as row player) simply calculates the row minimum of each row and then chooses the one with **largest row minimum**.

Similarly, N2 (as column player) can maximize its viewership (regardless of what N1 does) by calculating each column maximum and choosing column with the **smallest column maximum**.

It is easy to see that the two outcomes will satisfy the following inequality:

$$\max_{\text{all row}}(\text{row minimum}) \leq \min_{\text{all column}}(\text{column maximum})$$

In the example

	N2			
N1	Western	Action	Comedy	**MIN**
Western	35	15	60	15
Action	45	58	50	**45**
Comedy	38	14	70	14
MAX	**45**	58	50	

N1 chooses Action and N2 chooses Western. Then 45 million viewers will watch N1 and 55 million will watch N2; that this choice is simultaneously best for both networks. Now

$$\max_{\text{all row}}(\text{row minimum}) = \min_{\text{all column}}(\text{column maximum})$$

This is called **saddle point**, and the common value of both sides of the equation is called the **value of the game**. An **equilibrium** point of the game: choice of strategies for both players such that neither player can improve their payoff by changing his strategy.

In the above example each player's strategy was deterministic; they each examined possible outcomes and made a specific single choice to follow. This is called a **pure strategy**.

Theorem 9.1 *Let Σ_1 and Σ_2 is the set of strategies of Player1 (row) and Player2 (column), respectively, in a constant-sum game. A pure strategy pair $(\sigma, \tau) \in \Sigma_1 \times \Sigma_2$ is an equilibrium strategy if and only if the pair realizes the saddle point.*

9.1.2 Mixed strategies

On the other hand, there are games where following a pure strategy may not give the players the best outcome. Let us consider the following game.

1. Player 1 draws a French card from a card deck (and hides it from Player 2). She has the following options (i.e. strategies):

- **Pass**: she discards the card and pays $1 to Player 2.
- **Bet**: Player 2's turn follows.

2. Player 2 has the following options:
   - **Fold**: she pays $1 to Player 1.
   - **Call**: the card is revealed.
3. If the revealed card is high (10, Jack, Queen, King, Ace), then Player 2 pays $2 to Player 1. Otherwise, the card is low (2 through 9) and Player 1 pays $2 to Player 2.

Observe, that Player 1 can choose one of the following strategies:

- pass on both high and low card (**PP**)
- pass on high and bet on low (**PB**)
- bet on high and pass on low (**BP**)
- bet on both high and low (**BB**)

The possible expected outcomes of the game are then as follows.

	Player 2		
Player 1	**Call**	**Fold**	**MIN**
PP	-1	-1	-1
PB	-21/13	3/13	-21/13
BP	2/13	-3/13	-3/13
BB	-6/13	1	-6/13
MAX	2/13	1	

For example suppose that Player 1 plays **BP**, while Player 2 **calls**. The probability of getting a high card is $5/13$ and getting a low card is $8/13$. Therefore Player 1 expects $5/13 \cdot 2 + 8/13 \cdot (-1) = 2/13$ $.

This is a **zero-sum game** since either Player 1 pays Player 2 or vice-versa (the sum of the players' gains is zero). Observe that the game does not have a pure strategy saddle point.

Also notice that some strategies are better then others regardless of the other player's strategy. For instance, playing **BP** instead of **PP** always gives better outcome for Player 1. We say that a

strategy strongly (weakly) **dominates** another strategy if always gives better (or equal outcome). Clearly, if a strategy is dominated, it can be removed from the strategy set without changing the problem (its the optimal solution). After the possible simplifications, we obtain the following pay-off matrix:

	P2	
P1	**Call**	**Fold**
BP	2/13	-3/13
BB	-6/13	1

Instead of choosing a fix move, the players may consider to follow any strategy randomly according to a probability distribution. Now

- Player 1 chooses **BP** with a probability x_1 and **BB** with a probability x_2;
- clearly, $x_1, x_2 \geq 0$ and $x_1 + x_2 = 1$.
- The expected payoff if Player 2 **calls** is: $\frac{2}{13}x_1 - \frac{6}{13}x_2$,
- while if Player 2 **folds**, the expected payoff is $-\frac{3}{13}x_1 + x_2$.

The worst-case outcome for Player 1 is simply the minimum of the two:

$$\min_{(x_1,x_2)} \{\frac{2}{13}x_1 - \frac{6}{13}x_2, -\frac{3}{13}x_1 + x_2\}$$

Since $x_1 + x_2 = 1$, then we can simplify:

$$\text{payoff} = \min_{x_1}\{\frac{8}{13}x_1 - \frac{6}{13}, -\frac{16}{13}x_1 + 1\}$$

We can plot the possible payoffs based on x_1:

From this we determine the **best mixed strategy** for Player 1. That is the point E corresponds to strategy (x_1, x_2) where $x_1 = 19/24$ and $x_2 = 5/24$. The player's expected payoff is $1/39 \approx \$0.025$.

Similarly,

- Player 2 chooses **call** with a probability y_1 and **fold** with a probability y_2;
- clearly, $y_1, y_2 \geq 0$ and $y_1 + y_2 = 1$.

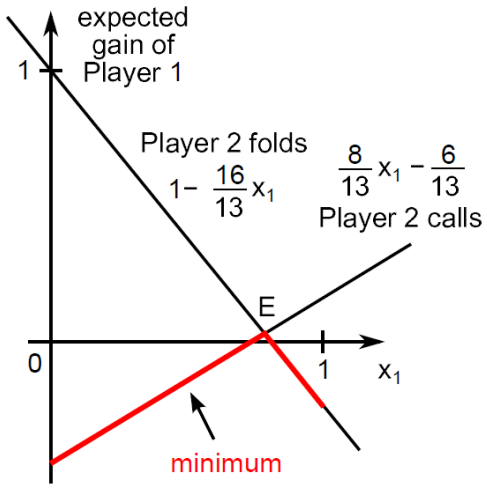


Figure 9.1: Source: Juraj Stacho's lecture notes.

- The expected payoff (loss) if Player 1 plays **BP** is: $\frac{2}{13}y_1 - \frac{3}{13}y_2$,
- while if Player 1 plays **BB**, the expected payoff (loss) is $-\frac{6}{13}y_1 + y_2$.

The worst-case outcome for Player 2 is the maximum of the two:

$$\max_{(y_1,y_2)}\{\frac{2}{13}y_1 - \frac{3}{13}y_2, -\frac{6}{13}y_1 + y_2\}$$

Since $y_1 + y_2 = 1$, thus

$$\text{payoff} = \min_{y1}\{\frac{5}{13}y_1 - \frac{3}{13}, -\frac{19}{13}x_1 + 1\}$$

We can plot the possible payoffs based on y_1:

From this we determine the **best mixed strategy** for Player 2. That is the point F corresponds to strategy (y_1, y_2) where $y_1 = 2/3$ and $y_2 = 1/3$. The Player 2's expected payoff is $-1/39 \approx \$ - 0.025$.

Theorem 9.2 (Mimimax, due to John Neumann) *Every zero-sum game has an equilibrium.*

9.2 Zero-sum games and LP

We could see that layer 1 tries to choose probabilities x_1, x_2 $(x_1 + x_2 = 1)$ such that maximize $\min_{(x_1,x_2)}\{\frac{2}{13}x_1 - \frac{6}{13}x_2, -\frac{3}{13}x_1 + x_2\}$; similarly, Player 2 chooses her probabilities y_1, y_2 $(y_1 + y_2 =$

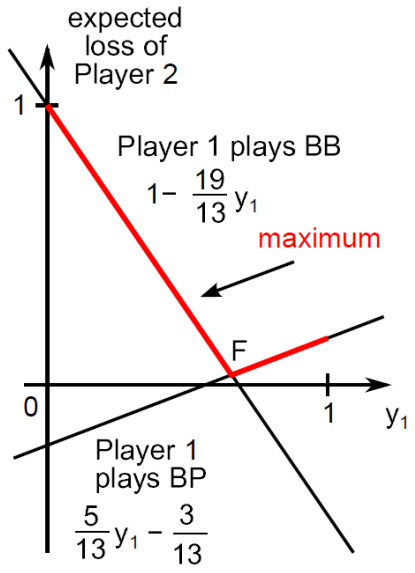


Figure 9.2: Source: Juraj Stacho's lecture notes.

1) such that minimize $\max_{(y_1,y_2)}\{\frac{2}{13}y_1 - \frac{3}{13}y_2, -\frac{6}{13}y_1 + y_2\}$. Both these problem can be transformed into linear programs as follows. For Player 1 the LP is

$$\begin{array}{lrcrcl} \text{Max} & z & & & & \\ \text{S.t.} & \frac{2}{13}x_1 & - & \frac{6}{13}x_2 & \geq & z \\ & -\frac{2}{13}x_1 & + & x_2 & \geq & z \\ & x_1 & + & x_2 & = & 1 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

and for Player 2

$$\begin{array}{lrcrcl} \text{Min} & w & & & & \\ \text{S.t.} & \frac{2}{13}y_1 & - & \frac{3}{13}y_2 & \leq & w \\ & -\frac{6}{13}y_1 & + & y_2 & \leq & w \\ & y_1 & + & y_2 & = & 1 \\ & & & y_1, y_2 & \geq & 0 \end{array}$$

After rewriting we get

$$
\begin{array}{lrcrcrcl}
\text{Max} & & & & & z & & \\
\text{S.t.} & -\frac{2}{13}x_1 & + & \frac{6}{13}x_2 & + & z & \leq & 0 \\
 & \frac{2}{13}x_1 & - & x_2 & + & z & \leq & 0 \\
 & x_1 & + & x_2 & & & = & 1 \\
 & & & & x_1, & x_2 & \geq & 0 \\
 & & & & & z & & \text{unrestricted}
\end{array}
$$

and

$$
\begin{array}{lrcrcrcl}
\text{Max} & & & & & w & & \\
\text{S.t.} & -\frac{2}{13}x_1 & + & \frac{3}{13}y_2 & + & w & \geq & 0 \\
 & \frac{6}{13}y_1 & - & y_2 & + & w & \geq & 0 \\
 & y_1 & + & y_2 & & & = & 1 \\
 & & & & y_1, & y_2 & \geq & 0 \\
 & & & & & w & & \text{unrestricted}
\end{array}
$$

Notice that the two programs are **duals** of each other. It is always the case in zero-sum (constant-sum) games. It tells us that the optimal solutions to the two programs have the same value by **strong duality** theorem (the result dates back to Gale and Tucker). The optimal solution is the value of the game. Moreover, the optimal strategies for the two players satisfy **complementary slackness**. The solutions to the two problems form an **equilibrium point** (neither play can do better by changing his/hers strategy). In the literature, this is a special case of the so-called **Nash equilibrium**. Providing deeper insights to zero-sum games is beyond the scope of the lecture, we only refer the cited literature for further reading.

We close this section by highlighting the deep connection between zero-sum games and linear programming.

Theorem 9.3 (Gale-Tucker, Luce-Raiffa) *For each zero-sum game there is a linear programming problem whose solution yields an equilibrium of the game and for each linear programming problem there is a zero-sum game whose equilibrium solution yields an optimal solution to the LP.*

9.3 Non-zero sum games

In many real situations we find that the gains/losses of the players are not necessarily sum up to zero (constant). This happens, for instance, in cases where players who cooperate can gain more

together than by competing alone.

9.3.1 Prisoner's dilemma

The following example, called **Prisoner's Dilemma**, is a classic problem in Game Theory. Two prisoners, say Bonnie and Clyde commit a bank robbery. They stash the cash and are driving around wondering what to do next when they are pulled over and arrested for a weapons violation. The police suspect Bonnie and Clyde of the bank robbery, but do not have any hard evidence. They separate the prisoners and offer the following options to Bonnie:

(1) If neither Bonnie nor Clyde confess, they will go to prison for 1 year on the weapons violation.

(2) If Bonnie confesses, but Clyde does not, then Bonnie can go free while Clyde will go to jail for 10 years.

(3) If Clyde confesses and Bonnie does not, then Bonnie will go to jail for 10 years while Clyde will go free.

(4) If both Bonnie and Clyde confess, then they will go to jail for 5 years.

A similar offer is made to Clyde. The following payoff matrix describes the situation:

	Clyde	
Bonnie	**Confess**	**Don't confess**
Confess	(-5, -5)	(0, -10)
Don't confess	(-10, 0)	(-1, -1)

Here payoffs are given in negative years (for years lost to prison). Obviously, the payoffs do not sum up to the same value each time.

It is easy to show that the strategy Confess dominates Don't Confess for Bonnie, and also for Clyde. Then we can simplify the payoff matrix, and conclude that game has only one Nash equilibrium, in which both player confess. (Notice, that both not confessing is not an equilibrium, since either player can change his mind and confess and thus not go to jail, while the other player gets 10 years).

Observe, that the Nash-equilibrium is not best possibility for the players, since they both can improve their situation by both changing strategy to Don't confess. This leads to the concept **Pareto efficiency**, that we do not discuss in this lecture.

9.3.2 Hawk and Dove game

Non-zero sum games have also been used to model various situations in *Evolutionary Biology*[1]. An important example is the **hawk-dove game**. Given a species with two subtypes or morphs with different strategies. The **Hawk** first displays aggression, then escalates into a fight until it either wins or is injured (loses). The **Dove** first displays aggression, but if faced with major escalation runs for safety. If not faced with such escalation, the Dove attempts to share the resource they fighting for. The payoff matrix of the game given as follows.

	Dove	**Hawk**
Dove	(2, 2)	(0, 4)
Hawk	(4, 0)	(-3, -3)

Explanation of the values:

- The value of the resource is 4, the damage when loosing a fight is -10
- If a Hawk meets a Dove he gets the full resource 4 to himself, while Dove gets 0
- If a Hawk meets a Hawk: half the time he wins, half the time he loses, so his average outcome is then $1/2 \cdot 4 + 1/2 \cdot (-10) = -3$
- If a Dove meets a Dove both share the resource and get $4/2 = 2$

Let the proportion of Hawks in the population is x, while proportion of doves is $(1 - x)$.

- The expected gain of a hawk is

$$-3x + 4(1 - x) = 4 - 7x;$$

- the The expected gain of a dove is

$$2(1 - x) - 0x = 2 - 2x.$$

[1] A book we suggest to the reader is Sir John Maynard Smith: "Evolution and the Theory of Games" (1982)

- Equilibrium means that there is not worth to change behavior to any individual while the behavior of the others does not chance. Then we get

$$4 - 7x = 2 - 2x \Rightarrow x = \frac{2}{5}.$$

Consider now that two individuals, whose behavior can either be hawk-type or dove-type, meet and decide behavior independently. Notice that there is no optimal pure strategy, thus suppose that individual A follows the mixed strategy $(x, 1-x)$, while individual B plays the mixed strategy $(y, 1-y)$. The Nash-equilibrium can be found by solving

$$\max \begin{bmatrix} x & 1-x \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 4 & -3 \end{bmatrix} \begin{bmatrix} y \\ 1-y \end{bmatrix} = \max \mathbf{x}^T A \mathbf{y}, \quad \text{s.t. } x \geq 0$$

for individual A, and

$$\max \begin{bmatrix} x & 1-x \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} y \\ 1-y \end{bmatrix} = \max \mathbf{x}^T B \mathbf{y}, \quad \text{s.t. } y \geq 0$$

for individual B. Separately, if either x or y would be known, these are linear programs. The problem is that we do not know any of these values. To solve the problem leads us the theory of **non-linear optimization**, now as a special case called **quadratic programming**, that we have already seen in Ch. 8. The discussion of the theory is beyond the scope of this lecture, we only refer to cited literature.

Theorem 9.4 (Existence of equlibria, Nash, 1949) *Every (n-player) game has at least one Nash equilibrium.*

9.4 Exercises

9.4.1 What is best mixed strategy of the players in the zero-sum game given by the following payoff matrix.

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

What is the value of the game? Formulate the corresponding LP problems.

9.4.2 How to chose λ in order to find dominance and thus simplify the game given by the following payoff matrix:

$$\begin{bmatrix} \lambda & \lambda^2 \\ 1 & 2 \end{bmatrix}$$

9.4.3 What is best mixed strategy of the players in the zero-sum game given by the following payoff matrix.

$$\begin{bmatrix} 0 & 2 \\ t & 1 \end{bmatrix}$$

where t is a real number?

9.4.4 Suppose that two players play the *Rock-Scissors-Paper* game and who lose pay \$1 to the winner. In case of a draw nothing happens. Show that the unique mixed equilibrium strategy of the game is $(x_1, x_2, x_3) = (y_1, y_2, y_3) = (1/3, 1/3, 1/3)$

9.4.5 Two player, independently to each other, write a number from 1 to 100 to a paper, then compare them. If the difference is exactly 1, then the player who wrote the smaller number pays \$1 to the other player. If the difference is at least 2, then, conversely, the player who wrote the larger number pays \$2 to the other player. If the numbers are equal nothing happens. What is the best strategy for each player?

9.4.6 Give various real-life situations, where the Prisoner's dilemma game reflects well (applicable) the scenario.

Chapter 10

Efficiency

We have discussed various methods to solve optimization problems such as linear programs (simplex method), minimum spanning tree, shortest paths and flows in networks (Kruskal, Prim, Bellman, Dijkstra, Ford-Fulkerson), integer linear programs (branch-and-bound, cutting planes). We have seen that the same problems can be solved using different approaches (for instance, we can solve shortest paths using the simplex method, or Dijkstra's algorithm, or dynamic programming). In this section we compare these methods in a uniform way.

10.1 Analysis of efficiency

Firstly we discuss in rough numbers the number of steps (operations) of the different methods.

10.1.1 Simplex algorithm

The number of steps of the simplex algorithm is proportional to the number of **bases** (dictionaries) we go through. The Bland's rule, for instance, guarantee that the same basis is not counted twice during the iterations. In general there are n variables and m equations (after introducing slack variables), thus the number of iterations is at most the number of different bases, that is $\binom{n+m}{m}$.This is roughly n^m for small m, but for large m (say $m = n/2$), using the *Stirling-formula*, is around 2^n. But this is a very pessimistic estimate. Unfortunately, there are examples which exhibit this **worst-case** behavior.

Example. (Klee and Minty, 1972)

$$\max \sum_{j=1}^{n} 10^{n-j} x_j$$

$$2\sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1} \quad i = 1, 2, \ldots, n$$

$$x_j \geq 0 \quad j = 1, 2, \ldots, n$$

In a special case, when $n = 3$, it looks as follows.

$$\begin{array}{lrcrcrcr} \max & z = 100x_1 & + & 10x_2 & + & 1x_3 & & \\ st & x_1 & & & & & \leq & 1 \\ & 20x_1 & + & x_2 & & & \leq & 100 \\ & 200x_1 & + & 20x_2 & + & x_3 & \leq & 10000 \\ & x_1 & , & x_2 & , & x_3 & \geq & 0 \end{array}$$

If at each step the entering variable is the one with largest coefficient in z (classic pivot rule), then the Klee-Minty examples go through $2^n - 1$ bases before finding the optimum.

R. Jeroszlov showed that using **steepest ascent rule** (means that the entering variable is chosen to be the one that provides largest rate of increase in the objective function) the number of iterations can also be exponentially large.

Good news is that in practice, according to e.g. Dantzig observations, if $m < 50$ and $n + m < 200$ then the number of iterations is around $3m/2$ and rarely happens that more than $3m$ iterations is needed. Moreover, there are polynomial time algorithms to solve LP problems (see e.g L.G Khachiyan's *ellipsoid method*, N. Karmarkar's *projective method* and some so-called interior point methods). The discussion of these is beyond a scope of these lecture notes.

10.1.2 Integer programming

Branch-and-bound

Solving integer programming problems with branch-and-bound each sub-problem is a linear program (can be solved by simplex or other methods) and only bounds on variables change. Thus, the size of each LP is the same in each sub-problem. There are 2^n possible sub-problems, that is unavoidable in general.

Cutting planes

Using cutting planes for solving IP problems, at each point we have only one linear program (there are no sub-problems). Each step adds one new constraint and one new variable to the problem, so the linear program grows at each step. There are possibly 2^n steps before optimum reached (unavoidable in general). The performance could be much worse than branch-and-bound if the size of the LP becomes too big (note that with branch-and-bound the LP remains the same size in all sub-problems).

10.1.3 Network problems

Kruskal and Prim algorithms

Given a network with n nodes, m edges. First we need to order the edges to a list according to their weights. By *quick-sort* this can be done by $O(m \log m)$ steps. Guaranteeing that no cycle produced in any step requires less using efficient implementations.

Dijkstra algorithm

Given a network with n nodes, m edges and we are finding the shortest $s - v$ path. Each of the n nodes can be the source node s. Each step involves finding a smallest value $d(v)$ and updating other values $d(w)$. This is roughly $2n$ calculations. All together this is about $2n^2$. It can be improved to $O(m \log n)$ using special data structures.

Bellman algorithm

For a single source at step k we consider paths using k edges, thus we need at most m calculations in each step, and all together $O(nm)$ operations are needed.

Ford-Fulkerson algorithm

Given a network with n nodes, m edges. Each step constructs the residual network, finds an augmenting path and augments the flow. This is roughly $2(n + m)$ operations for each step. At most nm steps needed if shortest augmenting path is used (see Edmonds-Karp algorithm). That is altogether roughly n^2m operations. This can be improved to $O(nm)$ by additional tricks.

Solving transportation problem

It can be solved using the so-called **transportation simplex method** (did not discuss here, but it can be found in the cited literature). It takes $\approx nm$ iterations.

Solving assignment problem

he number of required steps can be shown to be at most $\sqrt{n}$, thus the number of operations needed is $O(\sqrt{n}m)$

10.2 Summary of complexity results

Given algorithm $\mathcal{A}$ that takes an input x (numbers), performs a certain number of operations (additions, multiplications) and produces an output $\mathcal{A}(x)$ (e.g. a number, an answer yes/no, etc.). The **time complexity** of $\mathcal{A}$ is the number of (elementary) operations performed. The **space complexity** $\mathcal{A}$ is the size of memory used during the computation. Let $f(n)$ be the **worst case time complexity** of $\mathcal{A}$ for inputs of size n. That is the maximum number of operations that $\mathcal{A}$ performs on any input of size n.

The following "classification", in terms of complexity, is often considered:

- **Fast** or practical algorithms: $f(n) \approx \log n$, $f(n) \approx n$ $f(n) \approx n \log n$
- **Efficient** algorithms: $f(n) \approx n^{\alpha}$, where $\alpha = 1.5, 2, \ldots, 4, \ldots$
- Algorithms for **hard** problems: $f(n) \approx 2^n$, $f(n) \approx n!$, $f(n) \approx n^n$, $\ldots$

The following table is summarize the complexity of the discussed problems. The input to each of the problems will consist of n numbers (or $n + m$ numbers, or nm numbers where $n \geq m$ in case of LP). Here, L denotes the number of bits needed to represent any of these numbers (in a typical computer using IEEE754 floating point numbers $L = 24, 53, 64, \text{or } 113$).

10.3 Solving LP with computer: AMPL

Practical mathematical programming means running some algorithmic method on a computer and printing the optimal solution. The full sequence of the method is usually the following.

Problem	Time complexity	Space complexity
Linear programming (simplex): n variables, m constraints	$2^O(n)$	$O(nm)$
Linear Programming (Interior point): n variables, m constraints	$O(n^3L)$	$O(nm)$
Shortest path (Dijkstra): n nodes, m edges	$O(m + n \log n)$	$O(n)$
Shortest path (Bellman): n nodes, m edges	$O(nm)$	$O(n)$
Minimum Spanning tree (Prim, Kruskal): n nodes, m edges	$O(m \log m)$	$O(m)$
Transportation problem: n nodes, m edges	$O(nm)$	
Assignment problem: n nodes, m edges	$O(\sqrt{n}m)$	
Maximum flow: n nodes, m edges	$O(nm) = O(n^3)$	$O(m)$
0-1 Knapsack (Integer LP): n items	$2^O(n)$	$O(n)$

1. Formulate a model, the abstract system of variables, objectives, and constraints that represent the general form of the problem to be solved.

2. Collect data that define a specific problem instance.

3. Generate a specific objective function and constraint equations from the model and data.

4. Solve the problem instance by running a program, or solver, to apply an algorithm that finds optimal values of the variables.

5. Refine the model and data as necessary, and repeat.

AMPL offers an interactive command environment for setting up and solving mathematical programming problems. A flexible interface enables several solvers to be available at once so a user can switch among solvers and select options that may improve solver performance. The AMPL web site, www.ampl.com, provides free "student" versions of AMPL and representative solvers can easily handle problems of a few hundred variables and constraints.

Consider first our classic toy example:

$$\begin{aligned} \text{Max} \quad z = 3x_1 + 2x_2 \\ x_1 \quad + x_2 \leq 80 \\ 2x_1 \quad + x_2 \leq 100 \\ x_1 \qquad \leq 40 \\ x_1, x_2 \geq 0 \end{aligned}$$

To use AMPL, we need to create the following text file with the mathematical program code.

```
##Toy example

## Define variables
var Soldier;  #number of toy soldiers will be produced
var Train;    #number of toy trains will be produced

##Define objective function
maximize profit:  3*Soldier + 2*Train;

##Add constraints
subject to
Paint:  Soldier + Train <= 80;
Wood:  2*Soldier + Train <= 100;
Demand:  Soldier <= 40;
```

Note the followings about the AMPL language:

- The # symbol indicates the start of a comment. Everything after that symbol is ignored.
- Variables must be declared using the `var` keyword.
- All lines of code must end with a semi-colon (;).
- The objective starts with `maximize` or `minimize`, a name, and a colon (:). After that, the objective statement appears.
- constraint list start with `subject to`, a name, and a semi-colon. After that, the equation or inequality appears that must end wiht a semi-colon (;).
- Names must be unique. A variable and a constraint cannot have the same name.
- AMPL is case sensitive. Keywords must be in lower case.

After the file is created, it should be saved with the extension .mod. Once the model file is successfully loaded, tell AMPL to run the model by typing:

```
solve;
```

Although it was easy to transform the previous LP into a format AMPL understands, it is clear that if the problem had more details, or changed frequently, it would be much harder. For this reason, we typically use a more general algebraic way of stating linear programming models. Consider the following:

Given:
n: the number of different products
m: the number of different resources
c_j: the profit after selling one unit of product i
a_{ij}: the number of unit of resources i needed to produce one unit of product j
b_i: the available quantity (units) of resource i
u_j: the upper bound (e.g. demand) on the number of product j

Variables: x_j: the number of units produced from product i

The goal is to maximize profit: $z = \sum_{i=1}^{n} c_j x_j$

Subject to: $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ $(i = 1, \ldots m)$ and $0 \leq x_j \leq u_j$ $(j = 1, \ldots n)$.

Clearly, if $n = 2$, $m = 3$, $c_1 = 3$, $c_2 = 2$, $a_{11} = 1$, $a_{12} = 1$, $\ldots$, $a_{32} = 0$, $b_1 = 80$, $b_2 = 100$, $b_3 = 40$, then this is exactly the LP model for the toy example. The AMPL code can be the following:

```
##Resource allocation

##Define parameters
param n;
param c{j in 1..n};
param b{i in 1..m};
param usage {c, b};
param u{j in 1..n};
```

```
##Define variables
var product{j in 1..n};

##Define objective function
maximize z:  sum{j in 1..n} c[j]*product[j];

##Constraints
subject to capacity {i in 1...m}:
sum{j in 1..n} usage[i,j]*product[j] <= b[i];

subject to ubound {j in 1...n}:
0<= product[j] <= u[j];
```

Note the followings AMPL syntax rules:

- Each parameter declaration starts with the keyword `param`;
- Indexed parameters are declared using the syntax *varname in range*. For example, $c_j, j = 1, \ldots, n$ is de declared as `param c{j in 1..n};`
- Indexed variables are declared in the same way, starting with the `var` keyword.
- Summations are written similarly: $\sum_{j=1}^{n}$ is written `sum{j in 1..n}`
- Variable and parameter names can be anything meaningful, made up of upper and lower case letters, digits, and underscores.

In addition to specifying the model, we also must specify the data. There are different ways to do that, here we show one possibility.

```
param n := 2;
param m := 3;
param c := 1 3 2 2;
param b := 1 80 2 100 3 40;
param usage:  1 2 :=
```

```
1 1 1
2 2 1
3 1 0
;
```

The data file should be saved with the extension .dat.

By adding the keyword `integer` to the var declaration, we can restrict the declared variable to integral values. Furthermore, by adding the keyword `binary` to the var declaration, we can restrict the declared variable to the values 0 and 1. For a deepest introduction to AMPL, see e.g. [].

10.4 Exercises

10.4.1 Solve the exercises in Ch. 1 and Ch. 2 using AMPL.

10.4.2 Formulate the diet problem in AMPL with parameters. Solve it using specific parameters.

10.4.3 Formulate the shortest path problem in AMPL with parameters. Solve it using a given weighted and directed graph G

10.4.4 Formulate the transportation problem in AMPL with parameters.

10.4.5 Solve the IP exercises of Ch. 4 using AMPL. Implement problems with Either-Or and If-Then constraints.

10.4.6 Solve some special cases of the Klee-Minty problem with different available solvers in AMPL. Compare the running times.

Bibliography

[1] CHVATAL, V., CHVATAL, V., ET AL. *Linear programming*. Macmillan, 1983.

[2] DANTZIG, G. B., AND THAPA, M. N. *Linear programming 1: introduction*. Springer, 2006.

[3] FOURER, R., GAY, D. M., AND KERNIGHAN, B. *Ampl*, vol. 117. Boyd & Fraser Danvers, MA, 1993.

[4] GRIFFIN, C. *Game Theory: Penn State Math 486 Lecture Notes*. Penn State University, 2012.

[5] KALL, P., WALLACE, S. W., AND KALL, P. *Stochastic programming*. Springer, 1994.

[6] LAWLER, E. L. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.

[7] PLUHÁR, A. *Operációkutatás 1.* Szegedi Tudományegyetem, http://www.inf.u-szeged.hu/~pluhar/oktatas/lp.pdf.

[8] PLUHÁR, A. *Operációkutatás 2.* Szegedi Tudományegyetem, ttp://www.inf.u-szeged.hu/~pluhar/oktatas/or2.pdf.

[9] PRÉKOPA, A. *Stochastic programming*, vol. 324. Springer Science & Business Media, 2013.

[10] SCHRIJVER, A. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[11] SHAPIRO, A., DENTCHEVA, D., AND RUSZCZYŃSKI, A. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[12] STACHO, J. *Introduction to operations research*. Columbia University, 2014.

[13] VANDERBEI, R. J. *Linear programming*. Springer, 2015.

[14] WINSTON, W. L., AND GOLDBERG, J. B. *Operations research: applications and algorithms*, vol. 3. Thomson Brooks/Cole Belmont, 2004.

DESCRIPTION OF THE SUBJSECT – MASTER LEVEL

Title: Application of linear programming	**Credits: 3+1**
Category of the subject: compulsory	
The ratio of the theoretical and practical character of the subject: 75-25 (credit%)	
Type of the course: lecture + practice seminar **The total number of the contact hours**: 28 + 14 (lecture + practice seminar) during the semester, **Language**: English Further features of the teaching methodology: motivation, explanation and joint discussions with the students. The practice seminars are mainly based on students' activity and instructions of the lecturer.	
Evaluation: end-year written exam + students' presentations (lecture), mid-term written tests (practice seminar), in-class activity (lecture + practice seminar)	
The term of the course: I. semester	
Prerequisites (if any): -	

Description of the subject

General description:
Linear programming is a large and widely-used field of optimization. Many practical problems in operations research and optimization can be expressed as linear programming problems. It has a wide range of practical applications in business, commerce and industry, and - in parallel - its theoretical development has been continuously receiving much attention. Today, this theory is being successfully applied to problems of capital budgeting, design of diets, conservation of resources, games of strategy, economic growth prediction, and transportation systems, etc.

Aim:
The aim of the course is to introduce basic and some advanced theory of linear programming, and to present and solve real-life problems that can be described as linear programs. Special attention is paid for network problems, optimization in finance and stochastic problems, game theoretical models. A brief introduction to problem solving using the programming language AMPL is also provided.

Topics of the Course

- Introduction to mathematical modeling using linear programming
- Simplex algorithm, two-phase simplex method
- Duality theory and its economic interpretation, dual simplex method
- Integer programming, the branch-and-bound method
- Network problems
- The transshipment problem and applications
- Stochastic problems
- Basics of game theory
- Efficiency of solving algorithms of the discussed problems
- Brief introduction to AMPL

Selected bibliography (**2-5**) (author, title, edition, ISBN)

VANDERBEI , R. J.: Linear programming. Springer, 2015, ISBN 978-1-4614-7630-6
WINSTON , W. L., AND GOLDBERG , J. B.: Operations research: applications and algorithms, vol. 3. Thomson Brooks/Cole Belmont, 2004, ISBN 0-534-38058-1
STACHO, J.: Introduction to operations research. Columbia University, NewYork, 2014 (lecture notes)

General competence (knowledge, skills, etc., KKK 8.) promoted by the subject

a) Knowledge

Students will
- be familiar with the basic definitions and theorems of linear programming (LP)
- know the terminology of linear programming and the main fields of its applications
- understand LP models of real-life problems
- be familiar with the most important LP solving methods and available solvers
- have a broader picture of the related topics and their connections with LP

b) Skills

Students will be able to
- design linear programming models to real-life problems
- read and understand scientific papers and job reports of the topic
- solve LP problem using various available solvers
- actively participate in R&D projects where mathematical modeling involved
- apply LP related methods and solving techniques independently
- interpret and present their work and results in a correct way

c) Attitude

Students will be
- open to cooperate with other researchers and working groups
- ready to understand the mathematical model of real-world problems, use a solving methodology and interpret the results
- interested in new results, techniques and methods
- interested in contribute to new scientific results and methods

d) Autonomy and responsibility

Students will
- be able to organize their work and the work of small research teams independently
- help his colleagues in the completion of the R&D projects
- build their own professional and scientific career consciously
- be able to present their work for a competent audience

Special competence promoted by the subject

Knowledge	**Skills**	**Attitude**	**Autonomy/responsibility**
Students will know the main definitions, theorems and applications of linear programing	Students will be able to understand complex LP models of scientific and real-life problems	Students will be open to study and apply new methods of the topic	Students will be able to design and apply the appropriate LP models independently
Students will know various solving methods of LP and non-LP problems	Students will be able to design and solve LP and non-LP models	Students will be ready to cooperate with colleagues in R&D and scientific projects	Students will be able to evaluate and interpret results independently and correctly.
Students will know the usage of the modeling language AMPL	Students will be able to understand and interpret results and perform sensitivity analysis of the solution	Students will be ready to design and perform a complex project from data collection through math modeling to result presentation	Students will be able to present their work (models and solutions) and project independently in a correct way
Students will know the methodology behind data based mathematical modeling	They will be able to perform computer simulations and visualize the results	Students will be interested in new models, algorithms and applications	
Students will know the most important related topics to LP such as network problems, game theory, financial modeling	Students will be able to join R&D and scientific projects where mathematical modeling involved		

Instructor of the course (name, position, scientific degree):
Dr. Csendes Tibor, full professor, PhD, Dr. London András István, assistant professor, PhD

Teachers (name, position, scientific degree):
Dr. London András István, assistant professor, PhD

This teaching material has been made at the University of Szeged, and supported by the European Union. Project identity number: EFOP-3.4.3-16-2016-00014

Author: András London, PhD
Lecturers: András Pluhár, PhD, Zoltán Kincses, PhD
ISBN: 978-963-306-676-8
University of Szeged
2019, Szeged, Hungary

University of Szeged, Hungary
Venue: H-6720 Szeged, Dugonics square 13.
www.u-szeged.hu
www.szechenyi2020.hu