ORMS 1020

Operations Research

with GNU Linear Programming Kit

Tommi Sottinen

tommi.sottinen@uwasa.fi
www.uwasa.fi/∼tsottine/orms1020

August 27, 2009

Contents

Preface

These lecture notes are for the course ORMS1020 "Operations Research" for fall 2009 in the University of Vaasa. The notes are a slightly modified version of the notes for the fall 2008 course ORMS1020 in the University of Vaasa.

The chapters, or sections of chapters, marked with an asterisk (*) may be omitted — or left for the students to read on their own time — if time is scarce.

The author wishes to acknowledge that these lecture notes are collected from the references listed in Bibliography, and from many other sources the author has forgotten. *The author claims no originality*, and hopes not to be sued for plagiarizing or for violating the sacred © laws.

Vaasa August 27, 2009 T. S.

Bibliography

[1] Rodrigo Ceron: *The GNU Linear Programming Kit, Part 1: Introduction to linear optimization*, Web Notes, 2006.
http://www-128.ibm.com/developerworks/linux/library/l-glpk1/.

[2] Matti Laaksonen: *TMA.101 Operaatioanalyysi*, Lecture Notes, 2005.
http://lipas.uwasa.fi/∼mla/orms1020/oa.html.

[3] Hamdy Taha: *Operations Research: An Introduction* (6th Edition), Prentice Hall, Inc, 1997.

[4] Wayne Winston: *Operations Research: Applications and Algorithms*, International ed edition, Brooks Cole, 2004.

Part I

Introduction

Chapter 1

On Operations Research

This chapter is adapted from Wikipedia article *Operations Research* and [4, Ch. 1].

1.1 What is Operations Research

Definitions

To define anything non-trivial — like beauty or mathematics — is very difficult indeed. Here is a reasonably good definition of Operations Research:

1.1.1 Definition. *Operations Research* (OR) is an interdisciplinary branch of applied mathematics and formal science that uses methods like mathematical modeling, statistics, and algorithms to arrive at optimal or near optimal solutions to complex problems.

Definition 1.1.1 is problematic: to grasp it we already have to know, e.g., what is formal science or near optimality.

From a practical point of view, OR can be defined as an art of optimization, i.e., an art of finding minima or maxima of some objective function, and — to some extend — an art of defining the objective functions. Typical objective functions are

- profit,
- assembly line performance,
- crop yield,
- bandwidth,
- loss,
- waiting time in queue,
- risk.

From an organizational point of view, OR is something that helps management achieve its goals using the scientific process.

The terms OR and Management Science (MS) are often used synonymously. When a distinction is drawn, management science generally implies a closer relationship to Business Management. OR also closely relates to Industrial Engineering. Industrial engineering takes more of an engineering point of view, and industrial engineers typically consider OR techniques to be a major part of their tool set. Recently, the term Decision Science (DS) has also be coined to OR.

More information on OR can be found from the INFORMS web page

`http://www.thescienceofbetter.org/`

(If OR is "the Science of Better" the OR'ists should have figured out a better name for it.)

OR Tools

Some of the primary tools used in OR are

- statistics,
- optimization,
- probability theory,
- queuing theory,
- game theory,
- graph theory,
- decision analysis,
- simulation.

Because of the computational nature of these fields, OR also has ties to computer science, and operations researchers regularly use custom-written software.

In this course we will concentrate on optimization, especially linear optimization.

OR Motto and Linear Programming

The most common OR tool is Linear Optimization, or Linear Programming (LP).

1.1.2 Remark. The "Programming" in Linear Programming is synonym for "optimization". It has — at least historically — nothing to do with computer-programming.

LP is the OR'ists favourite tool because it is

- simple,
- easy to understand,

- robust.

"Simple" means easy to implement, "easy to understand" means easy to explain (to you boss), and "robust" means that it's like the Swiss Army Knife: perfect for nothing, but good enough for everything.

Unfortunately, almost no real-world problem is really a linear one — thus LP is perfect for nothing. However, most real-world problems are "close enough" to linear problems — thus LP is good enough for everything. Example 1.1.3 below elaborates this point.

1.1.3 Example. Mr. Quine sells gavagais. He will sell one gavagai for 10 Euros. So, one might expect that buying x gavagais from Mr. Quine would cost — according to the linear rule — $10x$ Euros.

The linear rule in Example 1.1.3 may well hold for buying 2, 3, or 5, or even 50 gavagais. But:

- One may get a discount if one buys 500 gavagais.
- There are only 1,000,000 gavagais in the world. So, the price for 1,000,001 gavagais is $+\infty$.
- The unit price of gavagais may go up as they become scarce. So, buying 950,000 gavagais might be considerably more expensive than €9,500,000.
- It might be pretty hard to buy 0.5 gavagais, since half a gavagai is no longer a gavagai (gavagais are bought for pets, and not for food).
- Buying -10 gavagais is in principle all right. That would simply mean selling 10 gavagais. However, Mr. Quine would most likely not buy gavagais with the same price he is selling them.

1.1.4 Remark. You may think of a curve that would represent the price of gavagais better than the linear straight line — or you may even think as a radical philosopher and argue that there is no curve.

Notwithstanding the problems and limitations mentioned above, linear tools are widely used in OR according to the following motto that should — as all mottoes — be taken with a grain of salt:

OR Motto. *It's better to be quantitative and naïve than qualitative and profound.*

1.2 History of Operations Research*

This section is most likely omitted in the lectures. Nevertheless, you should read it — history gives perspective, and thinking is nothing but an exercise of perspective.

Prehistory

Some say that Charles Babbage (1791–1871) — who is arguably the "father of computers" — is also the "father of operations research" because his research into the cost of transportation and sorting of mail led to England's universal "Penny Post" in 1840.

OR During World War II

The modern field of OR arose during World War II. Scientists in the United Kingdom including Patrick Blackett, Cecil Gordon, C. H. Waddington, Owen Wansbrough-Jones and Frank Yates, and in the United States with George Dantzig looked for ways to make better decisions in such areas as logistics and training schedules.

Here are examples of OR studies done during World War II:

- Britain introduced the convoy system to reduce shipping losses, but while the principle of using warships to accompany merchant ships was generally accepted, it was unclear whether it was better for convoys to be small or large. Convoys travel at the speed of the slowest member, so small convoys can travel faster. It was also argued that small convoys would be harder for German U-boats to detect. On the other hand, large convoys could deploy more warships against an attacker. It turned out in OR analysis that the losses suffered by convoys depended largely on the number of escort vessels present, rather than on the overall size of the convoy. The conclusion, therefore, was that a few large convoys are more defensible than many small ones.

- In another OR study a survey carried out by RAF Bomber Command was analyzed. For the survey, Bomber Command inspected all bombers returning from bombing raids over Germany over a particular period. All damage inflicted by German air defenses was noted and the recommendation was given that armor be added in the most heavily damaged areas. OR team instead made the surprising and counter-intuitive recommendation that the armor be placed in the areas which were completely untouched by damage. They reasoned that the survey was biased, since it only included aircraft that successfully came back from Germany. The untouched areas were probably vital areas, which, if hit, would result in the loss of the aircraft.

- When the Germans organized their air defenses into the Kammhuber Line, it was realized that if the RAF bombers were to fly in a bomber stream they could overwhelm the night fighters who flew in individual cells directed to their targets by ground controllers. It was then a matter of calculating the statistical loss from collisions against the statistical

loss from night fighters to calculate how close the bombers should fly to minimize RAF losses.

1.3 Phases of Operations Research Study

Seven Steps of OR Study

An OR project can be split in the following seven steps:

Step 1: Formulate the problem The OR analyst first defines the organization's problem. This includes specifying the organization's objectives and the parts of the organization (or system) that must be studied before the problem can be solved.

Step 2: Observe the system Next, the OR analyst collects data to estimate the values of the parameters that affect the organization's problem. These estimates are used to develop (in Step 3) and to evaluate (in Step 4) a mathematical model of the organization's problem.

Step 3: Formulate a mathematical model of the problem The OR analyst develops an idealized representation — i.e. a mathematical model — of the problem.

Step 4: Verify the model and use it for prediction The OR analyst tries to determine if the mathematical model developed in Step 3 is an accurate representation of the reality. The verification typically includes observing the system to check if the parameters are correct. If the model does not represent the reality well enough then the OR analyst goes back either to Step 3 or Step 2.

Step 5: Select a suitable alternative Given a model and a set of alternatives, the analyst now chooses the alternative that best meets the organization's objectives. Sometimes there are many best alternatives, in which case the OR analyst should present them all to the organization's decision-makers, or ask for more objectives or restrictions.

Step 6: Present the results and conclusions The OR analyst presents the model and recommendations from Step 5 to the organization's decision-makers. At this point the OR analyst may find that the decision-makers do not approve of the recommendations. This may result from incorrect definition of the organization's problems or decision-makers may disagree with the parameters or the mathematical model. The OR analyst goes back to Step 1, Step 2, or Step 3, depending on where the disagreement lies.

Step 7: Implement and evaluate recommendation Finally, when the organization has accepted the study, the OR analyst helps in implementing the recommendations. The system must be constantly monitored and updated dynamically as the environment changes. This means going back to Step 1, Step 2, or Step 3, from time to time.

In this course we shall concentrate on Step 3 and Step 5, i.e., we shall concentrate on mathematical modeling and finding the optimum of a mathematical model. We will completely omit the in-between Step 4. That step belongs to the realm of statistics. The reason for this omission is obvious: The statistics needed in OR is way too important to be included as side notes in this course! So, any OR'ist worth her/his salt should study statistics, at least up-to the level of parameter estimization.

Example of OR Study

Next example elaborates how the seven-step list can be applied to a queueing problem. The example is cursory: we do not investigate all the possible objectives or choices there may be, and we do not go into the details of modeling.

1.3.1 Example. A bank manager wants to reduce expenditures on tellers' salaries while still maintaining an adequate level of customer service.

Step 1: The OR analyst describes bank's objectives. The manager's vaguely stated wish may mean, e.g.,

- The bank wants to minimize the weekly salary cost needed to ensure that the average waiting a customer waits in line is at most 3 minutes.
- The bank wants to minimize the weekly salary cost required to ensure that only 5% of all customers wait in line more than 3 minutes.

The analyst must also identify the aspects of the bank's operations that affect the achievement of the bank's objectives, e.g.,

- On the average, how many customers arrive at the bank each hour?
- On the average, how many customers can a teller serve per hour?

Step 2: The OR analyst observes the bank and estimates, among others, the following parameters:

- On the average, how many customers arrive each hour? Does the arrival rate depend on the time of day?

- On the average, how many customers can a teller serve each hour? Does the service speed depend on the number of customers waiting in line?

Step 3: The OR analyst develops a mathematical model. In this example a queueing model is appropriate. Let

$$\begin{aligned} W_q &= \text{Average time customer waits in line} \\ \lambda &= \text{Average number of customers arriving each hour} \\ \mu &= \text{Average number of customers teller can serve each hour} \end{aligned}$$

A certain mathematical queueing model yields a connection between these parameters:

$$W_q = \frac{\lambda}{\mu(\mu - \lambda)}. \tag{1.3.2}$$

This model corresponds to the first objective stated in Step 1.

Step 4: The analyst tries to verify that the model (1.3.2) represents reality well enough. This means that the OR analyst will estimate the parameter W_q, λ, and μ statistically, and then she will check whether the equation (1.3.2) is valid, or close enough. If this is not the case then the OR analyst goes either back to Step 2 or Step 3.

Step 5: The OR analyst will optimize the model (1.3.2). This could mean solving how many tellers there must be to make μ big enough to make W_q small enough, e.g. 3 minutes.

We leave it to the students to wonder what may happen in steps 6 and 7.

Chapter 2

On Linear Programming

This chapter is adapted from [2, Ch. 1].

2.1 Example towards Linear Programming

Very Naïve Problem

2.1.1 Example. Tela Inc. manufactures two product: #1 and #2. To manufacture one unit of product #1 costs €40 and to manufacture one unit of product #2 costs €60. The profit from product #1 is €30, and the profit from product #2 is €20.

The company wants to maximize its profit. How many products #1 and #2 should it manufacture?

The solution is trivial: There is no bound on the amount of units the company can manufacture. So it should manufacture infinite number of either product #1 or #2, or both. If there is a constraint on the number of units manufactured then the company should manufacture only product #1, and not product #2. This constrained case is still rather trivial.

Less Naïve Problem

Things become more interesting — and certainly more realistic — when there are restrictions in the resources.

2.1.2 Example. Tela Inc. in Example 2.1.1 can invest €40,000 in production and use 85 hours of labor. To manufacture one unit of product #1 requires 15 minutes of labor, and to manufacture one unit of product #2 requires 9 minutes of labor.

The company wants to maximize its profit. How many units of product #1 and product #2 should it manufacture? What is the maximized profit?

The rather trivial solution of Example 2.1.1 is not applicable now. Indeed, the company does not have enough labor to put all the €40,000 in product #1.

Since the profit to be maximized depend on the number of product #1 and #1, our decision variables are:

$$\begin{aligned} x_1 &= \text{number of product \#1 produced,} \\ x_2 &= \text{number of product \#2 produced.} \end{aligned}$$

So the situation is: We want to *maximize* (max)

$$\text{profit:} \qquad 30x_1 + 20x_2$$

subject to (s.t.) the constraints

$$\begin{array}{lrcrcr} \text{money:} & 40x_1 & + & 60x_2 & \leq & 40{,}000 \\ \text{labor:} & 15x_1 & + & 9x_2 & \leq & 5{,}100 \\ \text{non-negativity:} & & & x_1, x_2 & \geq & 0 \end{array}$$

Note the last constraint: $x_1, x_2 \geq 0$. The problem does not state this explicitly, but it's implied — we are selling products #1 and #2, not buying them.

2.1.3 Remark. Some terminology: The unknowns x_1 and x_2 are called *decision variables*. The function $30x_1+20x_2$ to be maximized is called the *objective function*.

What we have now is a *Linear Program* (LP), or a Linear Optimization problem,

$$\begin{array}{lrcrcr} \max z & = & 30x_1 & + & 20x_2 & \\ \text{s.t.} & & 40x_1 & + & 60x_2 & \leq & 40{,}000 \\ & & 15x_1 & + & 9x_2 & \leq & 5{,}100 \\ & & & & x_1, x_2 & \geq & 0 \end{array}$$

We will later see how to solve such LPs. For now we just show the solution. For decision variables it is optimal to produce no product #1 and thus put all

the resource to product #2 which means producing 566.667 number of product #2. The profit will then be €11,333.333. In other words, the optimal solution is

$$\begin{aligned} x_1 &= 0, \\ x_2 &= 566.667, \\ z &= 11{,}333.333. \end{aligned}$$

2.1.4 Remark. If it is not possible to manufacture fractional number of products, e.g. 0.667 units, then we have to reformulate the LP-problem above to an *Integer Program* (IP)

$$\begin{array}{lrcrcr} \max z & = & 30x_1 & + & 20x_2 & \\ \text{s.t.} & & 40x_1 & + & 60x_2 & \leq & 40{,}000 \\ & & 15x_1 & + & 9x_2 & \leq & 5{,}100 \\ & & & & x_1, x_2 & \geq & 0 \\ & & & & x_1, x_2 & \text{are integers} & \end{array}$$

We will later see how to solve such IPs (which is more difficult than solving LPs). For now we just show the solution:

$$\begin{aligned} x_1 &= 1, \\ x_2 &= 565, \\ z &= 11{,}330. \end{aligned}$$

In Remark 2.1.4 above we see the usefulness of the OR Motto. Indeed, although the LP solution is not practical if we cannot produce fractional number of product, the solution it gives is close to the true IP solution: both in terms of the value of the objective function and the location of the optimal point. We shall learn more about this later in Chapter 8.

2.2 Solving Linear Programs Graphically

From Minimization to Maximization

We shall discuss later in Chapter 5, among other things, how to transform a minimization LP into a maximization LP. So, you should skip this subsection and proceed to the next subsection titled "Linear Programs with Two Decision Variables" — unless you want to know the general, and rather trivial, duality between minimization and maximization.

Any minimization problem — linear or not — can be restated as a maximization problem simply by multiplying the objective function by -1:

2.2.1 Theorem. *Let $K \subset \mathbb{R}^n$, and let $g : \mathbb{R}^n \to \mathbb{R}$. Suppose*

$$w^* = \min_{\mathbf{x} \in K} g(\mathbf{x})$$

and $\mathbf{x}^* \in \mathbb{R}^n$ *is a point where the minimum* w^* *is attained. Then, if* $f = -g$ *and* $z^* = -w^*$*, we have that*

$$z^* \quad = \quad \max_{\mathbf{x} \in K} f(\mathbf{x}),$$

and the maximum z^* *is attained at the point* $\mathbf{x}^* \in \mathbb{R}^n$*.*

The mathematically oriented should try to prove Theorem 2.2.1. It's not difficult — all you have to do is to not to think about the constraint-set K or any other specifics, like the space $\mathbb{R}^n$, or if there is a unique optimum. Just think about the big picture! Indeed, Theorem 2.2.1 is true in the greatest possible generality: It is true whenever it makes sense!

Linear Programs with Two Decision Variables

We shall solve the following LP:

2.2.2 Example.

$$\begin{array}{lrcrcll}
\max z & = & 4x_1 & + & 3x_2 & & \\
\text{s.t.} & & 2x_1 & + & 3x_2 & \leq 6 & (1) \\
& & -3x_1 & + & 2x_2 & \leq 3 & (2) \\
& & & & 2x_2 & \leq 5 & (3) \\
& & 2x_1 & + & x_2 & \leq 4 & (4) \\
& & & & x_1, x_2 & \geq 0 & (5)
\end{array}$$

The LP in Example 2.2.2 has only two decision variables: x_1 and x_2. So, it can be solved graphically on a piece of paper like this one. To solve graphically LPs with three decision variables would require three-dimensional paper, for four decision variables one needs four-dimensional paper, and so forth.

Four-Step Graphical Algorithm

Step 1: Draw coordinate space Tradition is that x_1 is the horizontal axis and x_2 is the vertical axis. Because of the non-negativity constraints on x_1 and x_2 it is enough to draw the 1st quadrant (the NE-quadrant).

Step 2: Draw constraint-lines Each constraint consists of a line and of information (e.g. arrows) indicating which side of the line is feasible. To draw, e.g., the line (1), one sets the inequality to be the equality

$$2x_1 + 3x_2 \quad = \quad 6.$$

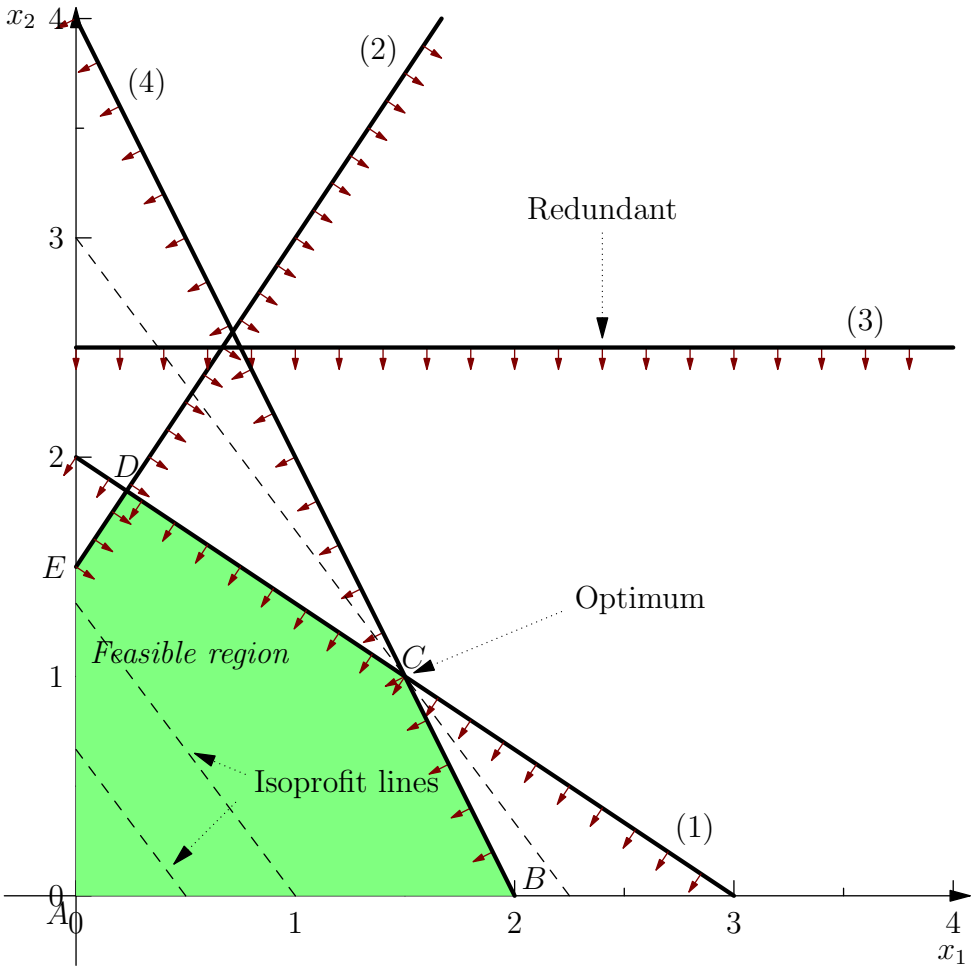
To draw this line we can first set $x_1 = 0$ and then set $x_2 = 0$, and we see that the line goes through points $(0, 2)$ and $(3, 0)$. Since (1) is a $\leq$-inequality, the feasible region must be below the line.

Step 3: Define feasible region This is done by selecting the region satisfied by all the constraints including the non-negativity constraints.

Step 4: Find the optimum by moving the isoprofit line The *isoprofit line* is the line where the objective function is constant. In this case the isoprofit lines are the pairs (x_1, x_2) satisfying

$$z \quad = \quad 4x_1 + 3x_2 \quad = \quad \text{const.}$$

(In the following picture we have drawn the isoprofit line corresponding to const $= 2$ and const $= 4$, and the optimal isoprofit line corresponding to const $= 9$.) The further you move the line from the origin the better value you get (unless the maximization problem is trivial in the objective function, cf. Example 2.2.3 later). You find the best value when the isoprofit line is just about to leave the feasible region completely (unless the maximization problem is trivial in constraints, i.e. it has an unbounded feasible region, cf. Example 2.2.4 later).



From the picture we read — by moving the isoprofit line away from the origin — that the optimal point for the decision variables (x_1, x_2) is

$$C \quad = \quad (1.5, 1).$$

Therefore, the optimal value is of the objective is

$$z \quad = \quad 4\times 1.5 + 3\times 1 \quad = \quad 9.$$

Example with Trivial Optimum

Consider the following LP maximization problem, where the objective function z does not grow as its arguments x_1 and x_2 get further away from the origin:

2.2.3 Example.

$$\begin{array}{lrcrcll}
\max z & = & -4x_1 & - & 3x_2 & & \\
\text{s.t.} & & 2x_1 & + & 3x_2 & \leq 6 & (1)\\
& & -3x_1 & + & 2x_2 & \leq 3 & (2)\\
& & & & 2x_2 & \leq 5 & (3)\\
& & 2x_1 & + & x_2 & \leq 4 & (4)\\
& & & & x_1, x_2 & \geq 0 & (5)
\end{array}$$

In this case drawing a graph would be an utter waste of time. Indeed, consider the objective function under maximization:

$$z = -4x_1 - 3x_2$$

Obviously, given the standard constraints $x_1, x_2 \geq 0$, the optimal solution is

$$\begin{aligned} x_1 &= 0,\\ x_2 &= 0,\\ z &= 0. \end{aligned}$$

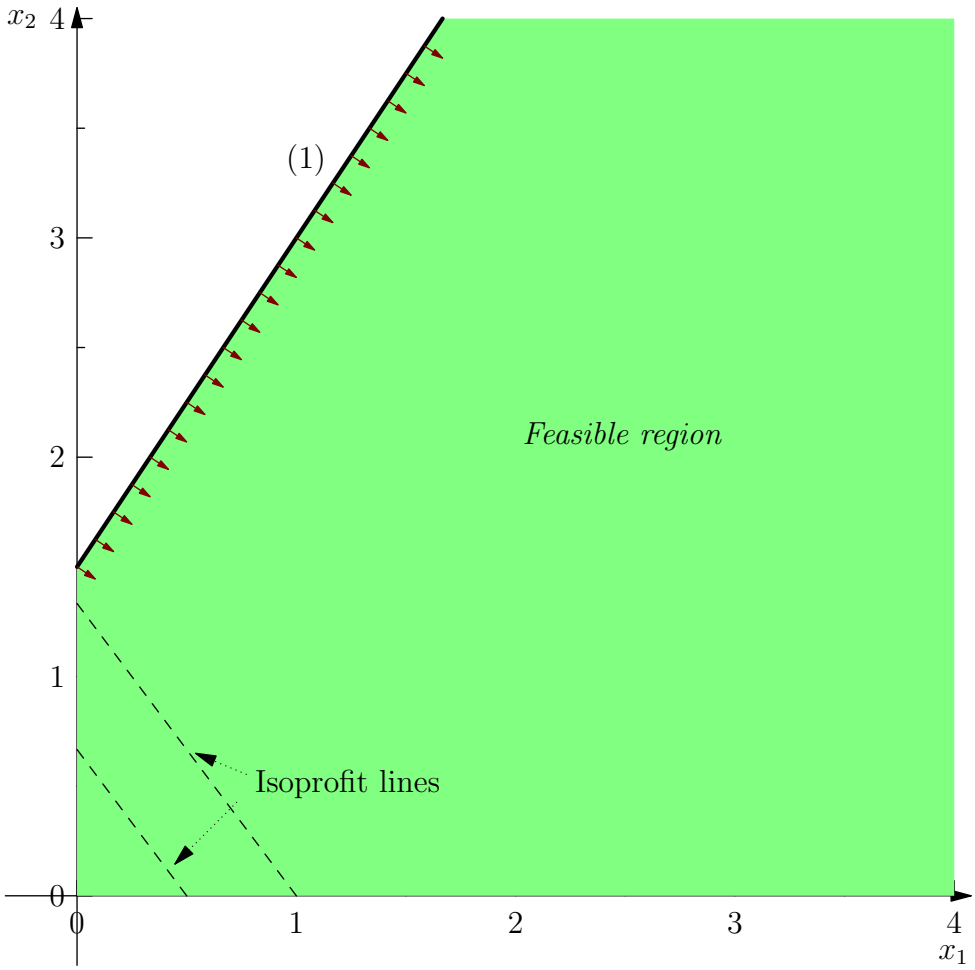
Whenever you have formulated a problem like this you (or your boss) must have done something wrong!

Example with Unbounded Feasible Region

2.2.4 Example.

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

From the picture below one sees that this LP has unbounded optimum, i.e., the value of objective function z can be made as big as one wishes.



Whenever you have formulated a problem like this you (or your boss) must have done something wrong — or you must be running a sweet business, indeed!

Chapter 3

Linear Programming with GNU Linear Programming Kit

This chapter is adapted from [1].

3.1 Overview of GNU Linear Programming Kit

GNU Linear Programming Kit

The GNU Linear Programming Kit (GLPK) is a software package intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. GLPK is written in ANSI C and organized as a callable library. GLPK package is part of the GNU Project and is released under the GNU General Public License (GPL).

The GLPK package includes the following main components:

- Revised simplex method (for LPs).
- Primal-dual interior point method (for LPs).
- Branch-and-bound method (for IPs).
- Translator for GNU MathProg modeling language.
- Application Program Interface (API).
- Stand-alone LP/MIP solver `glpsol`.

`glpsol`

GLPK is not a program — it's a library. GLPK can't be run as a computer program. Instead, client programs feed the problem data to GLPK through the GLPK API and receive results back.

However, GLPK has a default client: The `glpsol` program that interfaces with the GLPK API. The name "glpsol" comes from *G*NU *L*inear *P*rogram *Sol*ver. Indeed, usually a program like `glpsol` is called a solver rather than a client, so we shall use this nomenclature from here forward.

There is no standard Graphical User Interface (GUI) for `glpsol` that the author is aware of. So one has to call `glpsol` from a console. If you do not know how to use to a console in your Windows of Mac, ask your nearest guru now! Linux users should know how to use a console.

3.1.1 Remark. If you insist on having a GUI for GLPK, you may try

- http://bjoern.dapnet.de/glpk/ (Java GUI)
- http://qosip.tmit.bme.hu/∼retvari/Math-GLPK.html (Perl GUI)
- http://www.dcc.fc.up.pt/∼jpp/code/python-glpk/ (Python GUI)

The author does not know if any of these GUIs are any good.

To use `glpsol` we issue on a console the command

```
glpsol -m inputfile.mod -o outputfile.sol
```

or the command

```
glpsol -model inputfile.mod -output outputfile.sol
```

The commands above mean the same. Indeed, `-m` is an abbreviation to `-model`, and `-o` is an abbreviation to `-output`.

The option `-m inputfile.mod` tells the `glpsol` solver that the model to be solved is described in the file `inputfile.mod`, and the model is described in the GNU MathProg language. The option `-o outputfile.sol` tells the `glpsol` solver to print the results (the solution with some sensitivity information) to the file `outputfile.sol`.

GNU MathProg

The GNU MathProg is a modeling language intended for describing linear mathematical programming models.

Model descriptions written in the GNU MathProg language consists of:

- Problem decision variables.
- An objective function.
- Problem constraints.
- Problem parameters (data).

As a language the GNU MathProg is rather extensive, and can thus be a bit confusing. We shall not give a general description of the language in this course, but learn the elements of it through examples.

3.2 Getting and Installing GNU Linear Programming Kit

GLPK, like all GNU software, is open source: It is available to all operating systems and platforms you may ever use. This is the reason we use GLPK in this course instead of, say, LINDO.

General information on how to get GLPK and other GNU software can be found from

http://www.gnu.org/prep/ftp.html.

The GLPK source code can be downloaded from

ftp://ftp.funet.fi/pub/gnu/prep/glpk/.

If you use this method you have to compile GLPK by yourself. If you do not know what this means try following the instructions given in the links in one of the following subsections: Windows, Mac, or Linux.

Windows

From

http://gnuwin32.sourceforge.net/packages/glpk.htm

you should find a link to a setup program that pretty much automatically installs the GLPK for you. Some installation instructions can be found from

http://gnuwin32.sourceforge.net/install.html.

The instructions given above may or may not work with Windows Vista.

Mac

From

http://glpk.darwinports.com/

you find instruction how to install GLPK for Mac OS X.

Linux

If you are using Ubuntu then just issue the command

```
sudo apt-get install glpk
```

in the console (you will be prompted for your password). Alternatively, you can use the Synaptic Package Manager: Just search for glpk.

If you use a RedHat based system, consult

http://rpmfind.net/

with the keyword glpk.

Debian users can consult

http://packages.debian.org/etch/glpk.

For other Linux distributions, consult

http://www.google.com/.

3.3 Using `glpsol` with GNU MathProg

We show how to build up an LP model, how to describe the LP problem by using the GNU MathProg language, and how to solve it by using the `glpsol` program. Finally, we discuss how to interpret the results.

Giapetto's Problem

Consider the following classical problem:

3.3.1 Example. Giapetto's Woodcarving Inc. manufactures two types of wooden toys: soldiers and trains. A soldier sells for €27 and uses €10 worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by €14. A train sells for €21 and uses €9 worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by €10. The manufacture of wooden soldiers and trains requires two types of skilled labor: carpentry and finishing. A soldier requires 2 hours of finishing labor and 1 hour of carpentry labor. A train requires 1 hour of finishing and 1 hour of carpentry labor. Each week, Giapetto can obtain all the needed raw material but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldier are bought each week.

Giapetto wants to maximize weekly profits (revenues - costs).

To summarize the important information and assumptions about this problem:

1. There are two types of wooden toys: soldiers and trains.
2. A soldier sells for €27, uses €10 worth of raw materials, and increases variable labor and overhead costs by €14.

3. A train sells for €21, uses €9 worth of raw materials, and increases variable labor and overhead costs by €10.
4. A soldier requires 2 hours of finishing labor and 1 hour of carpentry labor.
5. A train requires 1 hour of finishing labor and 1 hour of carpentry labor.
6. At most, 100 finishing hours and 80 carpentry hours are available weekly.
7. The weekly demand for trains is unlimited, while, at most, 40 soldiers will be sold.

The goal is to find:

1. the numbers of soldiers and trains that will maximize the weekly profit,
2. the maximized weekly profit itself.

Mathematical Model for Giapetto

To model a linear problem (Giapetto's problem is a linear one — we will see this soon), the decision variables are established first. In Giapetto's shop, the objective function is the profit, which is a function of the amount of soldiers and trains produced each week. Therefore, the two decision variables in this problem are:

x_1: Number of soldiers produced each week
x_2: Number of trains produced each week

Once the decision variables are known, the objective function z of this problem is simply the revenue minus the costs for each toy, as a function of x_1 and x_2:

$$\begin{aligned} z &= (27 - 10 - 14)x_1 + (21 - 9 - 10)x_2 \\ &= 3x_1 + 2x_2. \end{aligned}$$

Note that the profit z depends linearly on x_1 and x_2 — this is a linear problem, so far (the constraints must turn out to be linear, too).

It may seem at first glance that the profit can be maximized by simply increasing x_1 and x_2. Well, if life were that easy, let's start manufacturing trains and soldiers and move to the Jamaica! Unfortunately, there are restrictions that limit the decision variables that may be selected (or else the model is very likely to be wrong).

Recall the assumptions made for this problem. The first three determined the decision variables and the objective function. The fourth and sixth assumption say that finishing the soldiers requires time for carpentry and finishing. The limitation here is that Giapetto does not have infinite carpentry and finishing hours. That's a constraint! Let's analyze it to clarify.

One soldier requires 2 hours of finishing labor, and Giapetto has at most 100 hours of finishing labor per week, so he can't produce more than 50 soldiers per

week. Similarly, the carpentry hours constraint makes it impossible to produce more than 80 soldiers weekly. Note here that the first constraint is stricter than the second. The first constraint is effectively a subset of the second, thus the second constraint is redundant.

The previous paragraph shows how to model optimization problems, but it's an incomplete analysis because all the necessary variables were not considered. It's not the complete solution of the Giapetto problem. So how should the problem be approached?

Start by analyzing the limiting factors first in order to find the constraints. First, what constrains the finishing hours? Since both soldiers and trains require finishing time, both need to be taken into account. Suppose that 10 soldiers and 20 trains were built. The amount of finishing hours needed for that would be 10 times 2 hours (for soldiers) plus 20 times 1 hour (for trains), for a total of 40 hours of finishing labor. The general constraint in terms of the decision variables is:

$$2x_1 + x_2 \quad \leq \quad 100.$$

This is a linear constraint, so we are still dealing with a linear program.

Now that the constraint for the finishing hours is ready, the carpentry hours constraint is found in the same way to be:

$$x_1 + x_2 \quad \leq \quad 80.$$

This constraint is a linear one.

There's only one more constraint remaining for this problem. Remember the weekly demand for soldiers? According to the problem description, there can be at most 40 soldiers produced each week:

$$x_1 \quad \leq \quad 40,$$

again a linear constraint.

The demand for trains is unlimited, so no constraint there.

The modeling phase is finished, and we have the following LP:

$$\begin{array}{lrcrcrl}
\max z & = & 3x_1 & + & 2x_2 & & & \text{(objective function)} \\
\text{s.t.} & & 2x_1 & + & x_2 & \leq & 100 & \text{(finishing constraint)} \\
& & x_1 & + & x_2 & \leq & 80 & \text{(carpentry constraint)} \\
& & x_1 & & & \leq & 40 & \text{(demand for soldiers)} \\
& & & & x_1, x_2 & \geq & 0 & \text{(sign constraints)}
\end{array}$$

Note the last constraint. It ensures that the values of the decision variables will always be positive. The problem does not state this explicitly, but it's still important (and obvious). The problem also implies that the decision variables are integers, but we are not dealing with IPs yet. So, we will just hope that

the optimal solution will turn out to be an integer one (it will, but that's just luck).

Now GLPK can solve the model (since GLPK is good at solving linear optimization problems).

Describing the Model with GNU MathProg

The mathematical formulation of Giapetto's problem needs to be written with the GNU MathProg language. The key items to declare are:

- The decision variables
- The objective function
- The constraints
- The problem data set

The following code, written in the (ASCII) text file `giapetto.mod`, shows how to solve Giapetto's problem with GNU MathProg. The line numbers are not part of the code itself. They have been added only for the sake of making references to the code.

```
 1  #
 2  # Giapetto's problem
 3  #
 4  # This finds the optimal solution for maximizing Giapetto's profit
 5  #
 6
 7  /* Decision variables */
 8  var x1 >=0;  /* soldier */
 9  var x2 >=0;  /* train */
10
11  /* Objective function */
12  maximize z: 3*x1 + 2*x2;
13
14  /* Constraints */
15  s.t. Finishing : 2*x1 + x2 <= 100;
16  s.t. Carpentry : x1 + x2 <= 80;
17  s.t. Demand    : x1 <= 40;
18
19  end;
```

Lines 1 through 5 are comments: # anywhere on a line begins a comment to the end of the line. C-style comments can also be used, as shown on line 7. They even work in the middle of a declaration. Comments are there to make the code more readable for a human reader. Computer reader, i.e. the GNU MathProg translator, will ignore comments.

Empty lines, like line 6, are simply ignored by the MathProg translator. So, empty lines can be used to structure the code more readable.

The first MathProg step is to declare the decision variables. Lines 8 and 9 declare `x1` and `x2`. A decision variable declaration begins with the keyword `var`. To simplify sign constraints, GNU MathProg allows a non-negativity constraint `>=0` in the decision variable declaration, as seen on lines 8 and 9.

Every sentence in GNU MathProg must end with a semicolon (`;`). Be careful! It is very easy to forget those little semicolons at the end of declarations! (Even moderately experienced programmers should be very familiar with this problem.)

Recall that `x1` represents soldier numbers, and `x2` represents train numbers. These variables could have been called `soldiers` and `trains`, but that would confuse the mathematicians in the audience. In general, it is good practice to use `x` for decision variables and `z` for the objective function. That way you will always spot them out quickly.

Line 12 declares the objective function. LP's can be either maximized or minimized. Giapetto's mathematical model is a maximization problem, so the keyword `maximize` is appropriate instead of the opposite keyword, `minimize`. The objective function is named `z`. It could have been named anything, e.g. `profit`, but this is not good practice, as noted before. Line 12 sets the objective function `z` to equal `3*x1+2*x2`. Note that:

- The colon (`:`) character separates the name of the objective function and its definition. Don't use equal sign (`=`) to define the objective function!
- The asterisk (`*`) character denotes multiplication. Similarly, the plus (`+`), minus (`-`), and forward slash (`/`) characters denote addition, subtraction, and division as you'd expect. If you need powers, then use either the circumflex (`^`) or the double-star (`**`).

Lines 15, 16, and 17 define the constraints. The keyword `s.t.` (subject to) is not required, but it improves the readability of the code (Remember good practice!). The three Giapetto constraints have been labeled `Finishing`, `Carpentry`, and `Demand`. Each of them is declared as in the mathematical model. The symbols `<=` and `>=` express the inequalities $\leq$ and $\geq$, respectively. Don't forget the `;` at the end of each declaration.

Every GNU MathProg text file must end with `end;`, as seen on line 19.

3.3.2 Remark. There was no data section in the code. The problem was so simple that the problem data is directly included in the objective function and constraints declarations as the coefficients of the decision variables in the declarations. For example, in the objective function, the coefficients 3 and 1 are part of the problem's data set.

Solving the Model with `glpsol`

Now, `glpsol` can use the text file `giapetto.mod` as input. It is good practice to use the `.mod` extension for GNU MathProg input files and redirect the solution

to a file with the extension `.sol`. This is not a requirement — you can use any file name and extension you like.

Giapetto's MathProg file for this example will be `giapetto.mod`, and the output will be in the text file `giapetto.sol`. Now, run `glpsol` in your favorite console:

```
glpsol -m giapetto.mod -o giapetto.sol
```

This command line uses two `glpsol` options:

- `-m` The `-m` or `-model` option tells `glpsol` that the input in the file `giapetto.mod` is written in GNU MathProg (the default modeling language for `glpsol`).
- `-o` The `-o` or `-output` option tells `glpsol` to send its output to the file `giapetto.sol`.

3.3.3 Remark. Some people prefer to use the extension `.txt` to indicate that the file in question is a (ASCII) text file. In that case `giapetto.mod` would be, e.g., `giapetto_mod.txt`. Similarly, `giapetto.sol` would be, e.g., `giapetto_sol.txt`. The command would be

```
glpsol -m giapetto_mod.txt -o giapetto_sol.txt
```

The solution report will be written into the text file `giapetto.sol` (unless you used the `.txt` extension style command of Remark 3.3.3), but some information about the time and memory GLPK consumed is shown on the system's standard output (usually the console window):

```
Reading model section from giapetto.mod...
19 lines were read
Generating z...
Generating Finishing...
Generating Carpentry...
Generating Demand...
Model has been successfully generated
glp_simplex: original LP has 4 rows, 2 columns, 7 non-zeros
glp_simplex: presolved LP has 2 rows, 2 columns, 4 non-zeros
lpx_adv_basis: size of triangular part = 2
*     0:   objval =   0.000000000e+00   infeas =   0.000000000e+00 (0)
*     3:   objval =   1.800000000e+02   infeas =   0.000000000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (114537 bytes)
lpx_print_sol: writing LP problem solution to 'giapetto.sol'...
```

The report shows that `glpsol` reads the model from the file `giapetto.mod` that has 19 lines, calls a GLPK API function to generate the objective function, and then calls another GLPK API function to generate the constraints.

After the model has been generated, `glpsol` explains briefly how the problem was handled internally by GLPK. Then it notes that an optimal solution is found. Then, there is information about the resources used by GLPK to solve the problem (Time used 0.0 secs, wow!). Finally the report tells us that the solution is written to the file `giapetto.sol`.

Now the optimal values for the decision variables `x1` and `x1`, and the optimal value of the objective function `z` are in the `giapetto.sol` file. It is a standard text file that can be opened in any text editor (e.g. Notepad in Windows, gedit in Linux with Gnome desktop). Here are its contents:

```
Problem:    giapetto
Rows:       4
Columns:    2
Non-zeros:  7
Status:     OPTIMAL
Objective:  z = 180 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            180
     2 Finishing    NU           100                       100             1
     3 Carpentry    NU            80                        80             1
     4 Demand       B             20                        40

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B             20             0
     2 x2           B             60             0

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err. = 0.00e+00 on column 0
        max.rel.err. = 0.00e+00 on column 0
        High quality

KKT.DB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

End of output
```

Interpreting the Results

The solution in the text file `giapetto.sol` is divided into four sections:

1. Information about the problem and the optimal value of the objective function.
2. Precise information about the status of the objective function and about the constraints.
3. Precise information about the optimal values for the decision variables.
4. Information about the optimality conditions, if any.

Let us look more closely:

Information about the optimal value of the objective function is found in the first part:

```
Problem:    giapetto
Rows:       4
Columns:    2
Non-zeros:  7
Status:     OPTIMAL
Objective:  z = 180 (MAXimum)
```

For this particular problem, we see that the solution is `OPTIMAL` and that Giapetto's maximum weekly profit is €180.

Precise information about the status of the objective function and about the constraints are found in the second part:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            180
     2 Finishing    NU           100                       100             1
     3 Carpentry    NU            80                        80             1
     4 Demand       B             20                        40
```

The Finishing constraint's status is `NU` (the `St` column). `NU` means that there's a non-basic variable (NBV) on the upper bound for that constraint. Later, when you know more operation research theory you will understand more profoundly what this means, and you can build the simplex tableau and check it out. For now, here is a a brief practical explanation: Whenever a constraint reaches its upper or lower boundary, it's called a bounded, or *active*, constraint. A bounded constraint prevents the objective function from reaching a better value. When that occurs, `glpsol` shows the status of the constraint as either `NU` or `NL` (for upper and lower boundary respectively), and it also shows the value of the marginal, also known as the *shadow price*. The marginal is the value by which the objective function would improve if the constraint were relaxed by one unit. Note that the improvement depends on whether the goal is to minimize or maximize the target function. For instance, in Giapetto's problem, which seeks maximization, the marginal value 1 means that the objective

function would increase by 1 if we could have one more hour of finishing labor (we know it's one more hour and not one less, because the finishing hours constraint is an upper boundary). The carpentry and soldier demand constraints are similar. For the carpentry constraint, note that it's also an upper boundary. Therefore, a relaxation of one unit in that constraint (an increment of one hour) would make the objective function's optimal value become better by the marginal value 1 and become 181. The soldier demand, however, is not bounded, thus its state is `B`, and a relaxation in it will not change the objective function's optimal value.

Precise information about the optimal values for the decision variables is found in the third part:

```
   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B              20             0
     2 x2           B              60             0
```

The report shows the values for the decision variables: `x1 = 20` and `x2 = 60`. This tells Giapetto that he should produce 20 soldiers and 60 trains to maximize his weekly profit. (The solution was an integer one. We were lucky: It may have been difficult for Giapetto to produce, say, 20.5 soldiers.)

Finally, in the fourth part,

```
Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err. = 0.00e+00 on column 0
        max.rel.err. = 0.00e+00 on column 0
        High quality

KKT.DB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality
```

there is some technical information mostly for the case when the algorithm is not sure if the optimal solution is found. In this course we shall omit this topic completely.

3.4 Advanced MathProg and `glpsol`*

This section is not necessarily needed in the rest of the course, and can thus be omitted if time is scarce.

`glpsol` Options

To get an idea of the usage and flexibility of `glpsol` here is the output of the command

```
glpsol -h
```

or the command

```
glpsol -help
```

which is the same command (`-h` is just an abbreviation to `-help`):

```
Usage: glpsol [options...] filename

General options:
   --mps             read LP/MIP problem in Fixed MPS format
   --freemps         read LP/MIP problem in Free MPS format (default)
   --cpxlp           read LP/MIP problem in CPLEX LP format
   --math            read LP/MIP model written in GNU MathProg modeling
                     language
   -m filename, --model filename
                     read model section and optional data section from
                     filename (the same as --math)
   -d filename, --data filename
                     read data section from filename (for --math only);
                     if model file also has data section, that section
                     is ignored
   -y filename, --display filename
                     send display output to filename (for --math only);
                     by default the output is sent to terminal
   -r filename, --read filename
                     read solution from filename rather to find it with
                     the solver
   --min             minimization
   --max             maximization
   --scale           scale problem (default)
   --noscale         do not scale problem
   --simplex         use simplex method (default)
   --interior        use interior point method (for pure LP only)
   -o filename, --output filename
                     write solution to filename in printable format
   -w filename, --write filename
                     write solution to filename in plain text format
   --bounds filename
                     write sensitivity bounds to filename in printable
                     format (LP only)
   --tmlim nnn       limit solution time to nnn seconds
   --memlim nnn      limit available memory to nnn megabytes
   --check           do not solve problem, check input data only
   --name probname   change problem name to probname
   --plain           use plain names of rows and columns (default)
```

```
   --orig            try using original names of rows and columns
                     (default for --mps)
   --wmps filename   write problem to filename in Fixed MPS format
   --wfreemps filename
                     write problem to filename in Free MPS format
   --wcpxlp filename write problem to filename in CPLEX LP format
   --wtxt filename   write problem to filename in printable format
   --wpb filename    write problem to filename in OPB format
   --wnpb filename   write problem to filename in normalized OPB format
   --log filename    write copy of terminal output to filename
   -h, --help        display this help information and exit
   -v, --version     display program version and exit

LP basis factorization option:
   --luf             LU + Forrest-Tomlin update
                     (faster, less stable; default)
   --cbg             LU + Schur complement + Bartels-Golub update
                     (slower, more stable)
   --cgr             LU + Schur complement + Givens rotation update
                     (slower, more stable)

Options specific to simplex method:
   --std             use standard initial basis of all slacks
   --adv             use advanced initial basis (default)
   --bib             use Bixby's initial basis
   --bas filename    read initial basis from filename in MPS format
   --steep           use steepest edge technique (default)
   --nosteep         use standard "textbook" pricing
   --relax           use Harris' two-pass ratio test (default)
   --norelax         use standard "textbook" ratio test
   --presol          use presolver (default; assumes --scale and --adv)
   --nopresol        do not use presolver
   --exact           use simplex method based on exact arithmetic
   --xcheck          check final basis using exact arithmetic
   --wbas filename   write final basis to filename in MPS format

Options specific to MIP:
   --nomip           consider all integer variables as continuous
                     (allows solving MIP as pure LP)
   --first           branch on first integer variable
   --last            branch on last integer variable
   --drtom           branch using heuristic by Driebeck and Tomlin
                     (default)
   --mostf           branch on most fractional varaible
   --dfs             backtrack using depth first search
   --bfs             backtrack using breadth first search
   --bestp           backtrack using the best projection heuristic
   --bestb           backtrack using node with best local bound
                     (default)
   --mipgap tol      set relative gap tolerance to tol
   --intopt          use advanced MIP solver
   --binarize        replace general integer variables by binary ones
                     (assumes --intopt)
   --cover           generate mixed cover cuts
```

```
   --clique          generate clique cuts
   --gomory          generate Gomory's mixed integer cuts
   --mir             generate MIR (mixed integer rounding) cuts
   --cuts            generate all cuts above (assumes --intopt)

For description of the MPS and CPLEX LP formats see Reference Manual.
For description of the modeling language see "GLPK: Modeling Language
GNU MathProg". Both documents are included in the GLPK distribution.

See GLPK web page at <http://www.gnu.org/software/glpk/glpk.html>.

Please report bugs to <bug-glpk@gnu.org>.
```

Using Model and Data Sections

Recall Giapetto's problem. It was very small. You may be wondering, in a problem with many more decision variables and constraints, would you have to declare each variable and each constraint separately? And what if you wanted to adjust the data of the problem, such as the selling price of a soldier? Do you have to make changes everywhere this value appears?

MathProg models normally have a model section and a data section, sometimes in two different files. Thus, `glpsol` can solve a model with different data sets easily, to check what the solution would be with this new data. The following listing, the contents of the text file `giapetto2.mod`, states Giapetto's problem in a much more elegant way. Again, the line numbers are here only for the sake of reference, and are not part of the actual code.

```
 1 #
 2 # Giapetto's problem (with data section)
 3 #
 4 # This finds the optimal solution for maximizing Giapetto's profit
 5 #
 6
 7 /* Set of toys */
 8 set TOY;
 9
10 /* Parameters */
11 param Finishing_hours {i in TOY};
12 param Carpentry_hours {i in TOY};
13 param Demand_toys     {i in TOY};
14 param Profit_toys     {i in TOY};
15
16 /* Decision variables */
17 var x {i in TOY} >=0;
18
19 /* Objective function */
20 maximize z: sum{i in TOY} Profit_toys[i]*x[i];
21
22 /* Constraints */
```

```
23 s.t. Fin_hours : sum{i in TOY} Finishing_hours[i]*x[i] <= 100;
24 s.t. Carp_hours : sum{i in TOY} Carpentry_hours[i]*x[i] <= 80;
25 s.t. Dem {i in TOY} : x[i] <= Demand_toys[i];
26
27
28 data;
29 /* data  section */
30
31 set TOY := soldier train;
32
33 param Finishing_hours:=
34 soldier         2
35 train           1;
36
37 param Carpentry_hours:=
38 soldier         1
39 train           1;
40
41 param Demand_toys:=
42 soldier        40
43 train    6.02E+23;
44
45 param Profit_toys:=
46 soldier         3
47 train           2;
48
49 end;
```

Rather than two separate files, the problem is stated in a single file with a modeling section (lines 1 through 27) and a data section (lines 28 through 49).

Line 8 defines a `SET`. A `SET` is a universe of elements. For example, if I declare mathematically `x`, for all `i` in `{1;2;3;4}`, I'm saying that `x` is an array, or vector, that ranges from 1 to 4, and therefore we have `x[1]`, `x[2]`, `x[3]`, `x[4]`. In this case, {1;2;3;4} is the set. So, in Giapetto's problem, there's a set called TOY. Where are the actual values of this set? In the data section of the file. Check line 31. It defines the TOY set to contain soldier and train. Wow, the set is not a numerical range. How can that be? GLPK handles this in an interesting way. You'll see how to use this in a few moments. For now, get used to the syntax for SET declarations in the data section:

```
set label := value1 value2 ...  valueN;
```

Lines 11, 12, and 13 define the parameters of the problem. There are three: `Finishing_hours`, `Carpentry_hours`, and `Demand_toys`. These parameters make up the problem's data matrix and are used to calculate the constraints, as you'll see later.

Take the `Finishing_hours` parameter as an example. It's defined on the `TOY` set, so each kind of toy in the `TOY` set will have a value for `Finishing_hours`. Remember that each soldier requires 2 hours of finishing work, and each train requires 1 hour of finishing work. Check lines 33,

34, and 35 now. There is the definition of the finishing hours for each kind of toy. Essentially, those lines declare that

`Finishing_hours[soldier]=2` and `Finishing_hours[train]=1`.

`Finishing_hours` is, therefore, a matrix with 1 row and 2 columns, or a row vector of dimension 2.

Carpentry hours and demand parameters are declared similarly. Note that the demand for trains is unlimited, so a very large value is the upper bound on line 43.

Line 17 declares a variable, `x`, for every `i` in `TOY` (resulting in `x[soldier]` and `x[train]`), and constrains them to be greater than or equal to zero. Once you have sets, it's pretty easy to declare variables, isn't it?

Line 20 declares the objective (target) function as the maximization of `z`, which is the total profit for every kind of toy (there are two: trains and soldiers). With soldiers, for example, the profit is the number of soldiers times the profit per soldier.

The constraints on lines 23, 24, and 25 are declared in a similar way. Take the finishing hours constraint as an example: it's the total of the finishing hours per kind of toy, times the number of that kind of toy produced, for the two types of toys (trains and soldiers), and it must be less than or equal to 100. Similarly, the total carpentry hours must be less than or equal to 80.

The demand constraint is a little bit different than the previous two, because it's defined for each kind of toy, not as a total for all toy types. Therefore, we need two of them, one for trains and one for soldiers, as you can see on line 25. Note that the index variable (`{i in TOY}`) comes before the `:`. This tells GLPK to create a constraint for each toy type in `TOY`, and the equation that will rule each constraint will be what comes after the `:`. In this case, GLPK will create

```
Dem[soldier] :  x[soldier] <= Demand[soldier]
Dem[train] :  x[train] <= Demand[train]
```

Solving this new model must yield the same results. So issue the command

```
glpsol -m giapetto2.mod -o giapetto2.sol
```

and the text file `giapetto2.sol` should read:

```
Problem:    giapetto2
Rows:       5
Columns:    2
Non-zeros:  8
Status:     OPTIMAL
Objective:  z = 180 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            180
     2 Fin_hours    NU           100                         100             1
```

```
     3 Carp_hours   NU            80                          80             1
     4 Dem[soldier] B             20                          40
     5 Dem[train]   B             60                    6.02e+23

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x[soldier]   B             20             0
     2 x[train]     B             60             0

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err. = 0.00e+00 on column 0
        max.rel.err. = 0.00e+00 on column 0
        High quality

KKT.DB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

End of output
```

Note how the constraints and the decision variables are now named after the `TOY` set, which looks clean and organized.

Part II

Theory of Linear Programming

Chapter 4

Linear Algebra and Linear Systems

Most students should already be familiar with the topics discussed in this chapter. So, this chapter may be a bit redundant, but it will at least serve us as a place where we fix some notation.

4.1 Matrix Algebra

Matrices, Vectors, and Their Transposes

4.1.1 Definition. A *matrix* is an array of numbers. We say that $\mathbf{A}$ is an $(m \times n)$-matrix if it has m rows and n columns:

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

Sometimes we write $\mathbf{A} = [a_{ij}]$ where it is understood that the index i runs from 1 through m, and the index j runs from 1 through n. We also use the denotation $\mathbf{A} \in \mathbb{R}^{m \times n}$ to indicate that $\mathbf{A}$ is an $(m \times n)$-matrix.

4.1.2 Example.

$$\mathbf{A} = \begin{bmatrix} 5 & 2 & -3 \\ 6 & 0 & 0.4 \end{bmatrix}$$

is a (2×3)-matrix, or $\mathbf{A} \in \mathbb{R}^{2 \times 3}$. E.g. $a_{12} = 2$ and $a_{23} = 0.4$; a_{32} does not exist.

4.1.3 Definition. The *transpose* $\mathbf{A}'$ of a matrix $\mathbf{A}$ is obtained by changing

its rows to columns, or vice versa: $a'_{ij} = a_{ji}$. So, if $\mathbf{A}$ is an $(m \times n)$-matrix

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

then its transpose $\mathbf{A}'$ is the $(n \times m)$-matrix

$$\mathbf{A}' \;=\; \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1m} \\ a'_{21} & a'_{22} & \cdots & a'_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a'_{n1} & a'_{n2} & \cdots & a'_{nm} \end{bmatrix} \;=\; \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}.$$

4.1.4 Example. If

$$\mathbf{A} \;=\; \begin{bmatrix} 5 & 2 & -3 \\ 6 & 0 & 0.4 \end{bmatrix},$$

then

$$\mathbf{A}' \;=\; \begin{bmatrix} 5 & 6 \\ 2 & 0 \\ -3 & 0.4 \end{bmatrix}.$$

So, e.g., $a'_{12} = 6 = a_{21}$.

4.1.5 Remark. If you transpose twice (or any even number of times), you are back where you started:

$$\mathbf{A}'' \;=\; (\mathbf{A}')' \;=\; \mathbf{A}.$$

4.1.6 Definition. A *vector* is either an $(n \times 1)$-matrix or a $(1 \times n)$-matrix. $(n \times 1)$-matrices are called *column vectors* and $(1 \times n)$-matrices are called *row vectors*.

4.1.7 Example. If $\mathbf{x}$ is the column vector

$$\mathbf{x} = \begin{bmatrix} 1 \\ -0.5 \\ -8 \\ 11 \end{bmatrix},$$

then $\mathbf{x}'$ is the row vector

$$\mathbf{x}' = [1 \quad -0.5 \quad -8 \quad 11].$$

4.1.8 Remark. We will always assume that vectors are column vectors. So, e.g., a 3-dimensional vector $\mathbf{x}$ will be

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

and not

$$[x_1 \quad x_2 \quad x_3].$$

Matrix Sums, Scalar Products, and Matrix Products

Matrix sum and scalar multiplication are defined component-wise:

4.1.9 Definition. Let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}.$$

Then the *matrix sum* $\mathbf{A} + \mathbf{B}$ is defined as

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

Let λ be a real number. Then the *scalar multiplication* $\lambda\mathbf{A}$ is defined as

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

4.1.10 Example. Let

$$\mathbf{A} = \begin{bmatrix} 5 & 2 \\ 33 & 20 \end{bmatrix} \quad \text{and} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then

$$\mathbf{A} - 100\mathbf{I} = \begin{bmatrix} -95 & 2 \\ 33 & -80 \end{bmatrix}.$$

4.1.11 Definition. Let $\mathbf{A}$ be a $(m \times n)$-matrix

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

and let $\mathbf{B}$ be a $(n \times p)$-matrix

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}.$$

Then the *product matrix* $[c_{ij}] = \mathbf{C} = \mathbf{AB}$ is the $(m \times p)$-matrix defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

4.1.12 Example.

$$\begin{bmatrix} 2 & 1 & 5 \\ 0 & 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & 5 & 3 & -1 \\ 7 & -4 & 2 & 5 \\ 0 & 2 & 1 & 6 \end{bmatrix} = \begin{bmatrix} 9 & 16 & 13 & 33 \\ 21 & -14 & 5 & 9 \end{bmatrix},$$

since, e.g.,

$$\begin{aligned} 9 &= c_{11} \\ &= a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \\ &= 2\times 1 + 1\times 7 + 5\times 0. \end{aligned}$$

4.1.13 Remark. Note that while matrix sum is commutative: $\mathbf{A}+\mathbf{B} = \mathbf{B}+\mathbf{A}$, the matrix product is not: $\mathbf{AB} \neq \mathbf{BA}$. Otherwise the matrix algebra follows the rules of the classical algebra of the real numbers. So, e.g.,

$$\begin{aligned}(\mathbf{A}+\mathbf{B})(\mathbf{C}+\mathbf{D}) &= (\mathbf{A}+\mathbf{B})\mathbf{C}+(\mathbf{A}+\mathbf{B})\mathbf{D}\\ &= \mathbf{AC}+\mathbf{BC}+\mathbf{AD}+\mathbf{BD}\\ &= \mathbf{A}(\mathbf{C}+\mathbf{D})+\mathbf{B}(\mathbf{C}+\mathbf{D}).\end{aligned}$$

Inverse Matrices

4.1.14 Definition. The *identity matrix* $\mathbf{I}_n$ is an $(n \times n)$-matrix (a square matrix) with 1s on the diagonal and 0s elsewhere:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

We shall usually write shortly $\mathbf{I}$ instead of $\mathbf{I}_n$, since the dimension n of the matrix is usually obvious.

4.1.15 Definition. The *inverse matrix* $\mathbf{A}^{-1}$ of a square matrix $\mathbf{A}$, if it exists, is such a matrix that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}.$$

4.1.16 Example. Let

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

Then

$$\mathbf{A}^{-1} = \begin{bmatrix} 3 & -2 \\ -1 & 1 \end{bmatrix}.$$

The inverse matrix $\mathbf{A}^{-1}$ in Example 4.1.16 above was found by using the Gauss–Jordan method that we learn later in this lecture. For now, the reader is invited to check that $\mathbf{A}^{-1}$ satisfies the two criteria of an inverse matrix: $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}$.

4.1.17 Example. Let

$$\mathbf{A} \;=\; \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

This matrix has no inverse. Indeed, if the inverse $\mathbf{A}^{-1}$ existed then, e.g., the equation

$$\mathbf{Ax} \;=\; \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

would have a solution in $\mathbf{x} = [x_1 \;\; x_2]'$:

$$\mathbf{x} \;=\; \mathbf{A}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

But this is impossible since

$$\mathbf{Ax} \;=\; \begin{bmatrix} x_1 + x_2 \\ 0 \end{bmatrix} \;\neq\; \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

no matter what $\mathbf{x} = [x_1 \;\; x_2]'$ you choose.

Dot and Block Matrix Notations

When we want to pick up rows or columns of a matrix $\mathbf{A}$ we use the dot-notation:

4.1.18 Definition. Let $\mathbf{A}$ be the matrix

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

Then its ith row is the n-dimensional row vector

$$\mathbf{a}_{i\bullet} \;=\; [a_{i1} \;\; a_{i2} \;\; \cdots \;\; a_{in}].$$

Similarly, $\mathbf{A}$'s jth column is the m-dimensional column vector

$$\mathbf{a}_{\bullet j} \;=\; \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}.$$

4.1.19 Example. If

$$\mathbf{A} = \begin{bmatrix} 5 & 2 & -3 \\ 6 & 0 & 0.4 \end{bmatrix},$$

then

$$\mathbf{a}_{2\bullet} = [6 \;\; 0 \;\; 0.4]$$

and

$$\mathbf{a}_{\bullet 3} = \begin{bmatrix} -3 \\ 0.4 \end{bmatrix}.$$

4.1.20 Remark. In statistical literature the dot-notation is used for summation: There $\mathbf{a}_{i\bullet}$ means the row-sum $\sum_{j=1}^{n} a_{ij}$. Please don't be confused about this. In this course the dot-notation does not mean summation!

When we want to combine matrices we use the block notation:

4.1.21 Definition. Let $\mathbf{A}$ be a $(m \times n)$-matrix

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

and let $\mathbf{B}$ be a $(m \times k)$-matrix

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mk} \end{bmatrix}.$$

Then the block matrix $[\mathbf{A} \;\; \mathbf{B}]$ is the $(m \times (n+k))$-matrix

$$[\mathbf{A} \;\; \mathbf{B}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1k} \\ a_{21} & a_{22} & \cdots & a_{2n} & b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_{m1} & b_{m2} & \cdots & b_{mk} \end{bmatrix}.$$

Similarly, if $\mathbf{C}$ is an $(p \times n)$-matrix, then the block matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix}$ is defined

as

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \\ c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \cdots & c_{pn} \end{bmatrix}.$$

4.1.22 Example. Let

$$\mathbf{A} = \begin{bmatrix} 5.1 & 2.1 \\ 6.5 & -0.5 \\ 0.1 & 10.5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 20 \\ 30 \end{bmatrix}, \quad \text{and} \quad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Then

$$\begin{bmatrix} 1 & -\mathbf{c}' \\ \mathbf{0} & \mathbf{A} \end{bmatrix} = \begin{bmatrix} 1 & -20 & -30 \\ 0 & 5.1 & 2.1 \\ 0 & 6.5 & -0.5 \\ 0 & 0.1 & 10.5 \end{bmatrix}.$$

Block matrices of the type that we had in Example 4.1.22 above appear later in this course when we solve LPs with the simplex method in lectures 6 and 7.

4.1.23 Example. By combining the dot and block notation we have:

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

Order Among Matrices

4.1.24 Definition. If $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ are both $(m \times n)$-matrices and $a_{ij} \leq b_{ij}$ for all i and j then we write $\mathbf{A} \leq \mathbf{B}$.

In this course we use the partial order introduced in Definition 4.1.24 mainly in the form $\mathbf{x} \geq \mathbf{b}$, which is then a short-hand for: $x_i \geq b_i$ for all i.

4.2 Solving Linear Systems

Matrices and linear systems are closely related. In this section we show how to solve a linear system by using the so-called Gauss–Jordan method. This is later important for us when we study the simplex method for solving LPs in lectures 6 and 7.

Matrices and Linear Systems

A *linear system* is the system of linear equations

$$\begin{array}{rcrcccrcl} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2, \\ & & & & & & & \vdots & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m. \end{array} \tag{4.2.1}$$

Solving the linear system (4.2.1) means finding the variables $x_1, x_2, \ldots, x_n$ that satisfy all the equations in (4.2.1) simultaneously.

The connection between linear systems and matrices is obvious. Indeed, let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

then the linear system (4.2.1) may be rewritten as

$$\mathbf{Ax} = \mathbf{b}.$$

Elementary Row Operations

We develop in the next subsection the Gauss–Jordan method for solving linear systems. Before studying it, we need to define the concept of an *Elementary Row Operation* (ERO). An ERO transforms a given matrix $\mathbf{A}$ into a new matrix $\tilde{\mathbf{A}}$ via one of the following operations:

ERO1 $\tilde{\mathbf{A}}$ is obtained by multiplying a row of $\mathbf{A}$ by a non-zero number λ:

$$\lambda \mathbf{a}_{\bullet i} \quad \rightsquigarrow \quad \tilde{\mathbf{a}}_{\bullet i}.$$

All other rows of $\tilde{\mathbf{A}}$ are the same as in $\mathbf{A}$.

ERO2 $\tilde{\mathbf{A}}$ is obtained by multiplying a row of $\mathbf{A}$ by a non-zero number λ (as in ERO1), and then adding some other row, multiplied by a non-zero number μ of $\mathbf{A}$ to that row:

$$\lambda \mathbf{a}_{\bullet i} + \mu \mathbf{a}_{\bullet j} \quad \rightsquigarrow \quad \tilde{\mathbf{a}}_{\bullet i}$$

($i \neq j$). All other rows of $\tilde{\mathbf{A}}$ are the same as in $\mathbf{A}$.

ERO3 $\tilde{\mathbf{A}}$ is obtained from $\mathbf{A}$ by interchanging two rows:

$$\begin{aligned} \mathbf{a}_{\bullet i} &\rightsquigarrow \tilde{\mathbf{a}}_{\bullet j} \\ \mathbf{a}_{\bullet j} &\rightsquigarrow \tilde{\mathbf{a}}_{\bullet i}. \end{aligned}$$

All other rows of $\tilde{\mathbf{A}}$ are the same as in $\mathbf{A}$.

4.2.2 Example. We want to solve the following linear system:

$$\begin{aligned} x_1 + x_2 &= 2, \\ 2x_1 + 4x_2 &= 7. \end{aligned}$$

To solve Example 4.2.2 by using EROs we may proceed as follows: First we replace the second equation by -2(first equation)$+$second equation by using ERO2. We obtain the system

$$\begin{aligned} x_1 + x_2 &= 2, \\ 2x_2 &= 3. \end{aligned}$$

Next we multiply the second equation by $1/2$ by using ERO1. We obtain the system

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 &= 3/2. \end{aligned}$$

Finally, we use ERO2: We replace the first equation by $-$(second equation)$+$first equation. We obtain the system

$$\begin{aligned} x_1 &= 1/2, \\ x_2 &= 3/2. \end{aligned}$$

We have solved Example 4.2.2: $x_1 = 1/2$ and $x_2 = 3/2$.

Now, let us rewrite what we have just done by using augmented matrix notation. Denote

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 7 \end{bmatrix},$$

and consider the augmented matrix

$$[\mathbf{A} \,|\, \mathbf{b}] = \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 4 & 7 \end{array}\right].$$

This is the matrix representation of the linear system of Example 4.2.2, and the three steps we did above to solve the system can be written in the matrix representation as

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 4 & 7 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 2 & 3 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1 & 3/2 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|c} 1 & 0 & 1/2 \\ 0 & 1 & 3/2 \end{array}\right]$$

The usefulness of EROs and the reason why the Gauss–Jordan methdod works come from the following theorem:

4.2.3 Theorem. *Suppose the augmented matrix* $[\tilde{\mathbf{A}}|\tilde{\mathbf{b}}]$ *is obtained from the augmented matrix* $[\mathbf{A}|\mathbf{b}]$ *via series of EROs. Then the linear systems* $\mathbf{Ax} = \mathbf{b}$ *and* $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$ *are equivalent.*

We shall not prove Theorem 4.2.3. However, Example 4.2.2 above should make Theorem 4.2.3 at least plausible.

Gauss–Jordan Method

We have already used the Gauss–Jordan method in this course — look how we solved Example 4.2.2. Now we present the general Gauss–Jordan method for solving $\mathbf{x}$ from $\mathbf{Ax} = \mathbf{b}$ as a three-step algorithm (steps 1 and 3 are easy; Step 2 is the tricky one):

Step 1 Write the problem $\mathbf{Ax} = \mathbf{b}$ in the augmented matrix form $[\mathbf{A}|\mathbf{b}]$.

Step 2 Using EROs transform the first column of $[\mathbf{A}|\mathbf{b}]$ into $[1\ 0\ \cdots\ 0]'$. This may require the interchange of two rows. Then use EROs to transform the second column of $[\mathbf{A}|\mathbf{b}]$ into $[0\ 1\ \cdots\ 0]'$. Again, this may require the change of two rows. Continue in the same way using EROs to transform successive columns so that ith column has the ith element equal to 1 and all other elements equal to 0. Eventually you will have transformed all the columns of the matrix $\mathbf{A}$, or you will reach a point where there are one or more rows of the form $[\mathbf{0}'|c]$ on the bottom of the matrix. In either case, stop and let $[\tilde{\mathbf{A}}|\tilde{\mathbf{b}}]$ denote the augmented matrix at this point.

Step 3 Write down the system $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$ and read the solutions $\mathbf{x}$ — or the lack of solutions — from that.

4.2.4 Remark.* the Gauss–Jordan method can also be used to calculate inverse matrices: To calculate the inverse of $\mathbf{A}$ construct the augmented matrix $[\mathbf{A}|\mathbf{I}]$ and transform it via EROs into $[\mathbf{I}|\mathbf{B}]$. Then $\mathbf{B} = \mathbf{A}^{-1}$.

4.3 Matrices as Linear Functions*

This section is not needed later in the course; it just explains a neat way of thinking about matrices. If you are not interested in thinking about matrices as functions you may omit this section.

Function Algebra

We only consider function between Euclidean spaces $\mathbb{R}^m$ although the definitions can be easily extended to any linear spaces.

4.3.1 Definition. Let $f, g : \mathbb{R}^n \to \mathbb{R}^m$, and let $\alpha \in \mathbb{R}$.

(a) The function *sum* $f + g$ is defined pointwise as

$$(f + g)(\mathbf{x}) \quad = \quad f(\mathbf{x}) + g(\mathbf{x}).$$

(b) The *scalar multiplication* αf is defined pointwise as

$$(\alpha f)(\mathbf{x}) \quad = \quad \alpha f(\mathbf{x}).$$

4.3.2 Example. Let $f : \mathbb{R}^2 \to \mathbb{R}$ map a point to its Euclidean distance from the origin and let $g : \mathbb{R}^2 \to \mathbb{R}$ project the point to its nearest point in the x_1-axis. So,

$$\begin{aligned} f(\mathbf{x}) &= \|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}, \\ g(\mathbf{x}) &= x_2. \end{aligned}$$

Then

$$(f + 2g)(\mathbf{x}) \quad = \quad \|x\| + 2x_2 \quad = \quad \sqrt{x_1^2 + x_2^2} + 2x_2.$$

So, e.g.,

$$(f + 2g)\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) \quad = \quad \sqrt{2} + 2.$$

4.3.3 Definition. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^k \to \mathbb{R}^n$. The *composition* $f \circ g$ is the function from $\mathbb{R}^k$ to $\mathbb{R}^m$ defined as

$$(f \circ g)(\mathbf{x}) \quad = \quad f(g(\mathbf{x})).$$

The idea of composition $f \circ g$ is: The mapping $f \circ g$ is what you get if you first map $\mathbf{x}$ through the mapping g into $g(\mathbf{x})$, and then map the result $g(\mathbf{x})$ through the mapping f into $f(g(\mathbf{x}))$.

4.3.4 Example. Let $g : \mathbb{R}^2 \to \mathbb{R}^2$ reflect the point around the x_1-axis, and let $f : \mathbb{R}^2 \to \mathbb{R}$ be the Euclidean distance of the point from the point $[1\ 2]'$. So,

$$\begin{aligned} g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) &= \left[\begin{array}{c} x_1 \\ -x_2 \end{array}\right], \\ f\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) &= \sqrt{(x_1-1)^2+(x_2-2)^2}. \end{aligned}$$

Then

$$\begin{aligned} (f\circ g)\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) &= f\left(g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right)\right) \\ &= f\left(\left[\begin{array}{c} x_1 \\ -x_2 \end{array}\right]\right) \\ &= \sqrt{(x_1-1)^2+(-x_2-2)^2}. \end{aligned}$$

So, e.g.,

$$(f\circ g)\left(\left[\begin{array}{c} 1 \\ 1 \end{array}\right]\right) = 3.$$

4.3.5 Definition. The *identity function* $\mathrm{id}_m : \mathbb{R}^m \to \mathbb{R}^m$ is defined by

$$\mathrm{id}_m(\mathbf{x}) = \mathbf{x}.$$

Sometimes we write simply id instead of id_m.

The idea of an identity map id is: If you start from $\mathbf{x}$, and map it through id, you stay put — id goes nowhere.

4.3.6 Definition. Let $f : \mathbb{R}^m \to \mathbb{R}^n$. If there exists a function $g : \mathbb{R}^n \to \mathbb{R}^m$ such that

$$g\circ f = \mathrm{id}_m \quad \text{and} \quad f\circ g = \mathrm{id}_n$$

then f is *invertible* and g is its *inverse function*. We denote $g = f^{-1}$.

The idea of an inverse function is: If you map $\mathbf{x}$ through f, then mapping the result $f(\mathbf{x})$ through f^{-1} gets you back to the starting point $\mathbf{x}$.

4.3.7 Example. Let $f : \mathbb{R} \to \mathbb{R}$ be $f(x) = x^3$. Then $f^{-1}(x) = \sqrt[3]{x}$. Indeed, now

$$f(f^{-1}(x)) \quad = \quad f\left(\sqrt[3]{x}\right) \quad = \quad \left(\sqrt[3]{x}\right)^3 \quad = \quad x,$$

and in the same way one can check that $f^{-1}(f(x)) = x$.

Matrix Algebra as Linear Function Algebra

Recall the definition of a linear function:

4.3.8 Definition. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *linear* if

$$f(\alpha\mathbf{x} + \beta\mathbf{y}) \quad = \quad \alpha f(\mathbf{x}) + \beta f(\mathbf{y}) \tag{4.3.9}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\alpha, \beta \in \mathbb{R}$.

The key connection between linear functions and matrices is the following theorem which says that for every linear function f there is a matrix $\mathbf{A}$ that defines it, and vice versa.

4.3.10 Theorem. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. The function f is linear if and only if there is a $(m \times n)$-matrix $\mathbf{A}$ such that*

$$f(\mathbf{x}) \quad = \quad \mathbf{A}\mathbf{x}.$$

Proof. The claim of Theorem 4.3.10 has two sides: (a) if a function f is defined by $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ then it is linear, and (b) if a function f is linear, then there is a matrix $\mathbf{A}$ such that $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$.

Let us first prove the claim (a). We have to show that (4.3.9) holds for f. But this follows from the properties of the matrix multiplication. Indeed,

$$\begin{aligned} f(\alpha\mathbf{x} + \beta\mathbf{y}) &= \mathbf{A}(\alpha\mathbf{x} + \beta\mathbf{y}) \\ &= \mathbf{A}(\alpha\mathbf{x}) + \mathbf{A}(\beta\mathbf{y}) \\ &= \alpha\mathbf{A}\mathbf{x} + \beta\mathbf{A}\mathbf{y} \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{y}). \end{aligned}$$

Let us then prove the claim (b). This is a bit more difficult than the claim (a), since mow we have to construct the matrix $\mathbf{A}$ from the function f. The trick is to write any vector $\mathbf{x} \in \mathbb{R}^n$ as

$$\mathbf{x} \quad = \quad \sum_{i=1}^{n} x_i \, \mathbf{e}_i,$$

where $\mathbf{e}_i = [e_{i1}\ e_{i2}\ \cdots\ e_{in}]'$ is the ith coordinate vector: $e_{ij} = 1$ if $i = j$ and 0 otherwise. Then, since f is linear, we have

$$f(\mathbf{x}) \quad = \quad f\left(\sum_{i=1}^{n} x_i\,\mathbf{e}_i\right) \quad = \quad \sum_{i=1}^{n} x_i\,f(\mathbf{e}_i)\,.$$

So, if we can write

$$\sum_{i=1}^{n} x_i\,f(\mathbf{e}_i) \quad = \quad \mathbf{A}\mathbf{x}$$

for some matrix $\mathbf{A}$ we are done. But this is accomplished by defining the matrix $\mathbf{A}$ by its columns as

$$\mathbf{a}_{\bullet i} \quad = \quad f(\mathbf{e}_i)\,.$$

This finishes the proof of Theorem 4.3.10. □

Since a row vector $[c_1\ c_2\ \cdots\ c_n]$ is an $(1\times n)$-matrix, we have the following corollary:

4.3.11 Corollary. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is linear if and only if there is a vector $\mathbf{c} \in \mathbb{R}^n$ such that*

$$f(\mathbf{x}) \quad = \quad \mathbf{c}'\mathbf{x}.$$

Finally, let us interpret the matrix operations as function operations.

4.3.12 Theorem. *Let $\mathbf{A}_h$ denote the matrix that corresponds to a linear function h. Let f and g be linear functions, and let λ be a real number. Then Âă*

$$\begin{aligned}
\mathbf{A}_{\lambda f} &= \lambda\mathbf{A}_f,\\
\mathbf{A}_{f+g} &= \mathbf{A}_f + \mathbf{A}_g,\\
\mathbf{A}_{f\circ g} &= \mathbf{A}_f\mathbf{A}_g,\\
\mathbf{A}_{\mathrm{id}} &= \mathbf{I},\\
\mathbf{A}_{f^{-1}} &= \mathbf{A}_f^{-1}.
\end{aligned}$$

The proof of Theorem 4.3.12 is omitted.

Chapter 5

Linear Programs and Their Optima

The aim of this chapter is, one the one hand, to give a general picture of LPs, and, on the other hand, to prepare for the Simplex method introduced in Chapter 6. If there is a one thing the student should remember after reading this chapter, that would be: *The optimal solution of an LP is found in one of the corners of the region of all feasible solutions.*

There are many theoretical results, i.e. theorems, in this lecture. The proofs of the theorems are collected in the last subsection, which is omitted in the course.

This lecture is adapted from [2, Ch. 2].

5.1 Form of Linear Program

Linear Program as Optimization Problem

Let us start by considering optimization in general. Optimization problems can be pretty diverse. The next definition is, for most practical purposes, general enough.

5.1.1 Definition. An *optimization problem* is: *maximize* (or *minimize*) the *objective function*

$$z \;=\; f(x_1, \ldots, x_n)$$

subject to the *constraints*

$$\begin{array}{ccccc} l_1 & \leq & g_1(x_1, \ldots, x_n) & \leq & u_1 \\ & & \vdots & & \\ l_m & \leq & g_m(x_1, \ldots, x_n) & \leq & u_m \end{array}$$

5.1.2 Remark. In some optimization problems some of the lower bounds $l_1, \ldots, l_m$ may be missing. In that case we may simply interpret the missing lower bounds to be $-\infty$. Similarly, some of the upper bounds $u_1, \ldots, u_m$ may be missing, and in that case we may interpret the missing upper bounds to be $+\infty$. Also, when one formulates an optimization problem, it may turn out

that the lower or upper bounds depend on the decision variables $x_1, \ldots, x_n$. In that case one can remove this dependence easily by using the following transformation:

$$\begin{array}{rcl} l_i(x_1,\ldots,x_n) & \leq & g_i(x_1,\ldots,x_n) \quad \leq \quad u_i(x_1,\ldots,x_n) \\ \rightsquigarrow & \left\{ \begin{array}{rcl} 0 & \leq & g_i(x_1,\ldots,x_n) - l_i(x_1,\ldots,x_n) \\ & & g_i(x_1,\ldots,x_n) - u_i(x_1,\ldots,x_n) \quad \leq \quad 0 \end{array} \right. \end{array}$$

So, the constraint i becomes two constraints, neither of which has bounds that depend on the decision $x_1, \ldots, x_n$.

The variables $x_1, \ldots, x_n$ in Definition 5.1.1 are called *decision variables*: They are the ones the optimizer seeks for — together with the value of the objective function, or course. Indeed, solving the optimization problem 5.1.1 means:

1. Finding the optimal decision $x_1^*, \ldots, x_n^*$ under which the objective $z^* = f(x_1^*, \ldots, x_n^*)$ is optimized — maximized or minimized, depending on the problem — among all possible decisions $x_1, \ldots, x_n$ that satisfy the constraints of the problem.
2. Finding, under the constraints, the optimal value $z^* = f(x_1^*, \ldots, x_n^*)$ — maximum or minimum, depending on the problem — of the objective.

It may look silly that we have split the solution criterion into two points, but sometimes it is possible to find the optimal value $z^* = f(x_1^*, \ldots, x_n^*)$ without finding the optimal decision $x_1^*, \ldots, x_n^*$ — or vice versa. In this course, however, we shall not encounter this situation.

5.1.3 Example. Mr. K. wants to invest in two stocks, #1 and #2. The following parameters have been estimated statistically:

$r_1 = 10\%$ is the return of stock #1,
$r_2 = 5\%$ is the return of stock #2,
$\sigma_1 = 4$ is the standard deviation of stock #1,
$\sigma_2 = 3$ is the standard deviation of stock #2,
$\rho = -0.5$ is the correlation between the stocks #1 and #2.

Mr. K. wants to maximize the return of his portfolio while keeping the risk (measured as standard deviation) of the portfolio below 3.5.

How should Mr. K. distribute his wealth between the two stocks?

Mr. K.'s problem in Example 5.1.3 is an optimization problem. Indeed, let w_1 and w_2 denote the portions of Mr. K.'s wealth put in stocks #1 and #2,

respectively. So, w_1 and w_2 are the decision variables of this problem. Then Mr. K.'s objective function to be maximized is the total return of his portfolio:

$$\begin{aligned} z &= f(w_1, w_2) \\ &= r_1 w_1 + r_2 w_2 \\ &= 10w_1 + 5w_2. \end{aligned}$$

The constraints are:

$$\begin{aligned} g_1(w_1, w_2) &= \sqrt{\sigma_1^2 w_1^2 + 2\rho\sigma_1\sigma_2 w_1 w_2 + \sigma_2^2 w_2} \\ &= \sqrt{16w_1^2 - 12w_1 w_2 + 9w_2^2} \\ &\leq 3.5, \end{aligned}$$

for the risk,

$$g_2(w_1, w_2) = w_1 + w_2 \leq 1,$$

for the total wealth to be invested, and — if short-selling is not allowed — then there are the sign constraints

$$\begin{aligned} 0 &\leq g_3(w_1, w_2) = w_1, \\ 0 &\leq g_4(w_1, w_2) = w_2. \end{aligned}$$

Let us then consider linear optimization problems.

Let us denote

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{l} = \begin{bmatrix} l_1 \\ \vdots \\ l_m \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}, \quad \text{and} \quad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ \vdots \\ g_m(\mathbf{x}) \end{bmatrix}.$$

Then Definition 5.1.1 can be written in a compact form as:

5.1.4 Definition. A *general optimization problem* is to either *maximize* or *minimize* the objective function

$$z = f(\mathbf{x})$$

subject to the constraints

$$\mathbf{l} \leq \mathbf{g}(\mathbf{x}) \leq \mathbf{u}.$$

An optimization problem is a linear optimization problem — or a linear program — if the objective function f and the constraint function $\mathbf{g}$ are both linear. Now, any linear form $\mathbf{h}(\mathbf{x})$ can be written as $\mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a matrix uniquely determined by the function $\mathbf{h}$ (if you want to see how to construct the matrix $\mathbf{A}$ from the function $\mathbf{h}$, see part (b) of the proof of Theorem 4.3.10). So, we arrive at the following definition:

5.1.5 Definition. A *linear optimization problem*, or a *linear program* (LP) is to either *maximize* or *minimize* the *objective function*

$$z \quad = \quad \mathbf{c}'\mathbf{x}$$

subject to the *constraints*

$$\mathbf{l} \quad \leq \quad \mathbf{A}\mathbf{x} \quad \leq \quad \mathbf{u},$$

and to the *sign constraints*

$$\mathbf{x} \quad \geq \quad \mathbf{0}.$$

5.1.6 Remark. The sign constraints $\mathbf{x} \geq \mathbf{0}$ in Definition 5.1.5 are somewhat particular (as opposed to general), and not in line with Definition 5.1.1. However, in practice the sign constraints are so prevalent that we make it a standing assumption.

Mr. K.'s optimization problem in Example 5.1.3 was not an LP, since the constraint function g_1 was not linear (everything else in Mr. K.'s problem was linear).

Assumptions of Linear Programs

Definition 5.1.5 is the mathematical description of an LP. As such it is complete and perfect, as is the nature of mathematical definitions. Definition 5.1.5 is also very Laconic and not directly related to the "real world", as is also the nature of mathematical definitions. The list below explains the consequences — or assumptions, if you like — of Definition 5.1.5 for the non-Spartans living in the "real world":

Proportionality The contribution to the objective function from each decision variable is proportional to the value of the decision variable: If, say, decision variable x_2 is increased by Δ then the value of objective function is increased by $c_2\Delta$. Similarly, the contribution of each decision variable in restrictions is also proportional to the value of the said variable. So, e.g., if you double the value of the decision variable x_2 the resources consumed by that decision will also double.

Additivity The contribution to the objective function for any variable is independent of the values of the other decision variables. For example, no matter what the value of x_1 is increasing x_2 to $x_2 + \Delta$ will increase the value of the objective function by $c_2\Delta$. Similarly, the resources used by decision x_2 will increase independently of the value of x_1.

Divisibility It is assumed that the decision variables can take fractional values. For example x_1 may be π. This assumption is in many practical cases not true, but a reasonably good approximation of the reality. In case this assumption is violated, we have an Integer Program (IP). We shall learn about IPs in Chapter 11.

Certainty It is assumed that all the parameters of the program are known with certainty. For example, in Giapetto's problem 3.3.1 it was assumed that the demands for soldiers and trains were known. This is certainly almost never the case in practice. Indeed, typically in practice one has to estimate the parameters of the LP statistically. We shall not talk about statistical estimation in this course.

5.1.7 Remark. Unlike proportionality, additivity, and divisibility, the certainty assumption is not particularly "linear".

Standard Form of Linear Programs

The LP 5.1.5 can be represented in many equivalent forms. In this course we consider three forms:

1. standard form,
2. slack form,
3. canonical slack form.

The standard form is good for theoretical considerations. The slack form and the canonical slack form are food (no typo here) for the Simplex algorithm. In this subsection we consider the standard form. The slack form and the canonical form will be introduced in Chapter 6 where we study the Simplex algorithm.

5.1.8 Remark. When you solve LPs with GLPK there is usually no need to transform them to standard, slack, or canonical forms: GLPK will internally transform the LP into any form that is suitable for it (which is probably some kind of a slack form as GLPK uses a revised Simplex algorithm). Also, note that there is no universal consensus on what is a "standard form", or a "slack form", or a "canonical slack form" of an LP. So, in different textbooks you are likely to find different definitions. Indeed, e.g. [4] calls the slack form a standard form.

5.1.9 Definition. A *standard form* LP is:

$$\begin{array}{lrcl} \max & z & = & \mathbf{c}'\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

So, an LP is in standard form if:

1. It is a maximization problem
2. There are no lower bound constraints

Any LP of Definition 5.1.5 can be transformed into a standard form LP of Definition 5.1.9 by using the following three-step algorithm:

Step 1: Change into maximization If the LP is a minimization problem, change it to a maximization problem by multiplying the objective vector $\mathbf{c}$ by -1:

$$\min\ \mathbf{c}'\mathbf{x} \quad \rightsquigarrow \quad \max\ -\mathbf{c}'\mathbf{x}.$$

Step 2: Remove double inequalities If there are both lower and upper bound in a single constraint, change that constraint into two constraints:

$$\begin{aligned} l_i \le\ & a_{i1}x_1 + \cdots + a_{in}x_n \ \le u_i \\ \rightsquigarrow\ & \begin{cases} l_i \le\ a_{i1}x_1 + \cdots + a_{in}x_n \\ \qquad\quad a_{i1}x_1 + \cdots + a_{in}x_n \ \le u_i \end{cases} . \end{aligned}$$

Step 3: Remove lower bounds If there is a lower bound constraint l_i, change it to an upper bound constraint by multiplying the corresponding inequality by -1:

$$l_i \ \le\ a_{i1}x_1 + \cdots a_{in}x_n \quad \rightsquigarrow \quad -a_{i1}x_1 - \cdots - a_{in}x_n \ \le\ -l_i.$$

5.1.10 Example. Let us find the standard form of the LP

$$\min z \ = \ -2x_1 + 3x_2$$

$$\begin{array}{lrcrcrcll} \text{s.t.} & 1 & \le & x_1 & + & x_2 & \le & 9 & (1) \\ & & & 2x_1 & - & x_2 & \le & 4 & (2) \\ & 2 & \le & 7x_1 & + & x_2 & \le & 100 & (3) \\ & & & & & x_1, x_2 & \ge & 0 & (4) \end{array}$$

Step 1: We turn the LP into a maximization problem, and get the objective

$$\max z \ = \ 2x_1 - 3x_2.$$

Step 2: We remove the double inequalities (1) and (3). From the constraint (1) we get the constraints

$$\begin{array}{rcrcrcll} 1 & \le & x_1 & + & x_2 & & & (1.a) \\ & & x_1 & + & x_2 & \le & 9 & (1.b) \end{array}$$

and from the constraint (3) we get the constraints

$$\begin{array}{rcrcrcll} 2 & \le & 7x_1 & + & x_2 & & & (3.a) \\ & & 7x_1 & + & x_2 & \le & 100 & (3.b) \end{array}$$

Before going to Step 3 let us check the status of the LP now:

$$\max z \ = \ -2x_1 + 3x_2$$

$$
\begin{array}{lrcrcrcrl}
\text{s.t.} & 1 & \leq & x_1 & + & x_2 & & & (1.a) \\
 & & & x_1 & + & x_2 & \leq & 9 & (1.b) \\
 & & & 2x_1 & - & x_2 & \leq & 4 & (2) \\
 & 2 & \leq & 7x_1 & + & x_2 & & & (3.a) \\
 & & & 7x_1 & + & x_2 & \leq & 100 & (3.b) \\
 & & & & & x_1, x_2 & \geq & 0 & (4)
\end{array}
$$

Step 3: We remove the lower bounds for the inequalities (1.a) and (3.a). We obtain the standard form

$$
\begin{array}{lrcrcrcrl}
 & \max z & = & -2x_1 + 3x_2 & & & & & \\
\text{s.t.} & -x_1 & - & x_2 & \leq & -1 & & & (1.a) \\
 & x_1 & + & x_2 & \leq & 9 & & & (1.b) \\
 & 2x_1 & - & x_2 & \leq & 4 & & & (2) \\
 & -7x_1 & - & x_2 & \leq & -2 & & & (3.a) \\
 & 7x_1 & + & x_2 & \leq & 100 & & & (3.b) \\
 & & & x_1, x_2 & \geq & 0 & & & (4)
\end{array}
$$

5.2 Location of Linear Programs' Optima

In this section we consider the region of the admissible decisions in an LP problem — the so-called feasible region. We also consider the location of the optimal decision of an LP, which must of course be in the feasible region.

The main result — and a problem — to be remembered is:

The optimal solution of an LP is found in *one of the corners* of the feasible region. The decision variables corresponding to the corners are called *Basic Feasible Solutions* (BFS). So, *the problem is to find the best BFS.*

Shape of Feasible Region

Since we now know how to transform any LP into a standard form, we shall state LPs in their standard forms in the definitions.

5.2.1 Definition. The *feasible region* K of an LP

$$
\begin{array}{lcl}
\max z & = & \mathbf{c}'\mathbf{x} \\
\text{s.t.} & & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
 & & \mathbf{x} \geq \mathbf{0}
\end{array}
$$

is the set of decisions $\mathbf{x}$ that satisfy the constraints $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$:

$$
K = \{\mathbf{x} \in \mathbb{R}^n \,;\, \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}\,.
$$

Note that the feasible region K is determined by the technology matrix $\mathbf{A}$ and the constraints $\mathbf{b}$. The objective $\mathbf{c}$ has no effect on the feasible region.

5.2.2 Remark. In what follows we use the convention that LP and its feasible region are like in Definition 5.2.1, and that the said LP has n decision variables and m constraints, excluding the sign constraints. This means that $\mathbf{c}$ is an n-dimensional column vector, $\mathbf{A}$ is an $(n \times m)$-matrix, and $\mathbf{b}$ is an m-dimensional column vector.

Theorem 5.2.3 below says that if you have two feasible solutions, and you draw a line segment between those two solutions, then every solution in that line segment is also feasible.

5.2.3 Theorem. *The feasible region of an LP is convex: If* $\mathbf{x}$ *and* $\mathbf{y}$ *belong to the feasible region, then also* $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ *belong to the feasible region for all* $\alpha \in [0, 1]$.

5.2.4 Remark.* Actually the feasible region has more structure than just convexity: It is a (closed) *convex polytope*. We shall not give a general definition of a convex polytope. If the (closed) convex polytope is bounded one can think it as the result of the following procedure:

1. Take some points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k$ to be corners of the convex polytope. These points belong to the convex polytope — they are its generators.
2. If some point, say $\mathbf{q}$, is in a line segment connecting any two points of the convex polytope, then $\mathbf{q}$ must be included to the convex polytope also. This including procedure must be reiterated as new points are included into the set until the set there are no more new points to be included.

For example, three points will generate a filled triangle as their convex polytope. The said three points will be the three corners of the polytope. Four points in a three-dimensional space will generate a filled (irregular) tetrahedron as their convex polytope with the said four points at its corners.

Optima in Corners

5.2.5 Definition. Consider a feasible solution, or a decision, $\mathbf{x}$ of an LP. The constraint b_i is *active* at the decision $\mathbf{x}$ if $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$.

Constraint i being active at decision $\mathbf{x}$ means that the resource i is fully consumed, or utilized, with decision $\mathbf{x}$.

5.2.6 Definition. A feasible solution, or decision, of an LP is

Inner point if there are no active constraints at that decision,
Boundary point if there is at least one active constraints at that decision,

Corner point if there are at least n linearly independent active constraints at that decision. Corner points are also called *Basic Feasible Solutions* (BFS).

Note that corner points are also boundary points, but not vice versa.

5.2.7 Remark. Linear independence means that the constraints are genuinely different. For example, the constraints

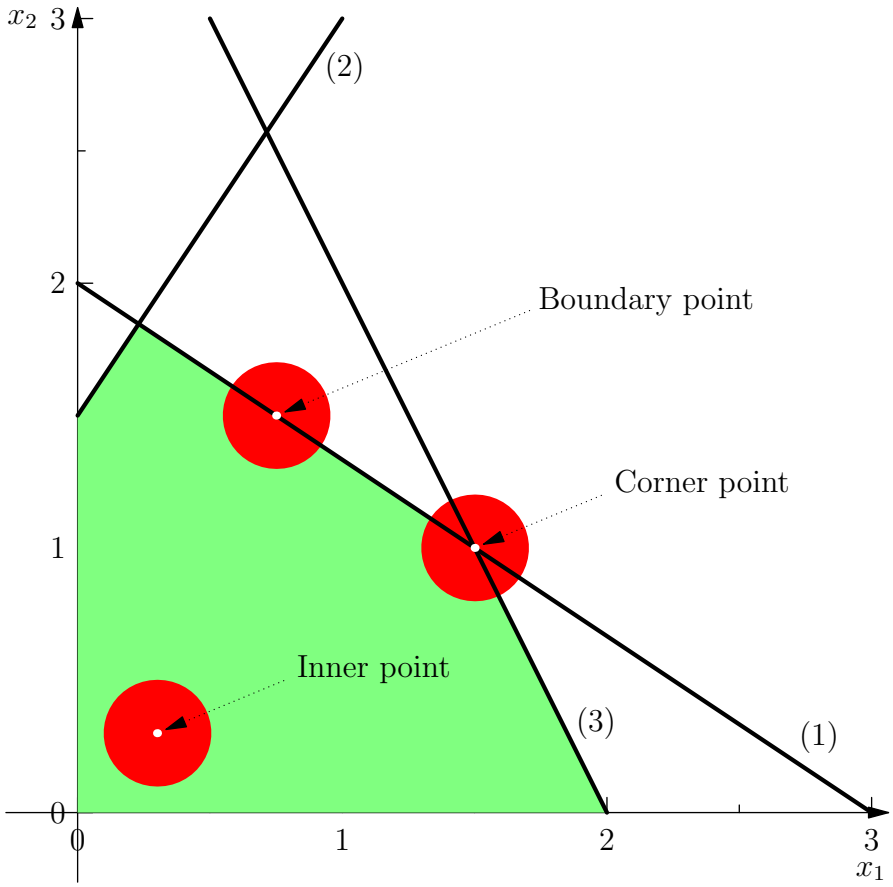
$$\begin{array}{rcrcl} 2x_1 & + & 3x_2 & \leq & 2 \\ x_1 & + & x_2 & \leq & 4 \end{array}$$

are linearly independent, but the constraints

$$\begin{array}{rcrcl} 2x_1 & + & 3x_2 & \leq & 2 \\ 6x_1 & + & 9x_2 & \leq & 6 \end{array}$$

are not.

The next picture illustrates Definition 5.2.6. In that picture: None of the constraints (1), (2), or (3) is active in the "Inner point". In the "Boundary point" one of the constraints, viz. (1), is active. In the "Corner point" two (which is the number of the decision variables) of the constraints, viz. (1) and (3), are active.



5.2.8 Theorem. *Let* $\mathbf{x}^*$ *be an optimal solution to an LP. Then* $\mathbf{x}^*$ *is a boundary point of the feasible region.*

Theorem 5.2.8 can be refined considerably. Indeed, the next theorem tells us that in seeking the optimum we do not have to check the entire boundary — it is enough to check the corners!

5.2.9 Theorem. *An optimal solution* $\mathbf{x}^*$ *of an LP can be found — when it exists — in one of the corner points of the feasible region, i.e., an optimal solution is a BFS.*

5.3 Karush–Kuhn–Tucker Conditions*

Sometimes one can make an educated guess about the optimal corner of an LP. In that case one asks if the guess is correct. The following Karush–Kuhn–Tucker theorem provides a way to check the correctness of one's guess.

5.3.1 Theorem. *Consider the LP*

$$\begin{array}{lcrcl} \max z & = & \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & & \mathbf{A}\mathbf{x} & \leq & \mathbf{b}\,. \\ & & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

Let $\mathbf{x}$ *be a BFS of the LP. Suppose there are vectors* $\mathbf{s}, \mathbf{u}, \mathbf{v}$ *such that*

(i) $\mathbf{A}\mathbf{x} + \mathbf{s} = \mathbf{b}$,
(ii) $\mathbf{c} = \mathbf{A}'\mathbf{v} - \mathbf{u}$,
(iii) $\mathbf{u}'\mathbf{x} + \mathbf{v}'\mathbf{s} = 0$,
(iv) $\mathbf{s}, \mathbf{u}, \mathbf{v} \geq \mathbf{0}$.

Then $\mathbf{x}$ *is an optimal solution to the LP.*

The vectors $\mathbf{s}$, $\mathbf{u}$, $\mathbf{v}$ in the Karush–Kuhn–Tucker theorem 5.3.1 have the following interpretation:

- $\mathbf{s}$ is the slack vector: s_i tells how much of the resource i is unused. If $s_i = 0$ then the constraint i is active, i.e., the resource i is completely used. This interpretation is obvious if you look condition (i) of Theorem 5.3.1.
- $\mathbf{u}$ is connected to the sign constraint $\mathbf{x} \geq 0$: if $x_i > 0$ then the ith sign constraint is not active and $u_i = 0$.
- $\mathbf{v}$ is connected to the resource constraints. If there is slack $s_i > 0$ in the resource i then $v_i = 0$.

5.3.2 Remark. The `KKT.PE`, `KKT.PB`, `KKT.DE`, and `KKT.DB` in the `glpsol`'s report are related to the conditions (i), (ii), (iii), and (iv) of the Karush–Kuhn–Tucker theorem 5.3.1. If the Karush–Kuhn–Tucker conditions are satisfied the values in the `glpsol`'s Karush–Kuhn–Tucker section should all be zero.

5.4 Proofs*

Proof of Theorem 5.2.3. Let $\alpha \in [0, 1]$. It is obvious that if $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{y} \geq \mathbf{0}$, then also $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \geq \mathbf{0}$. So, it remains to show that if $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{A}\mathbf{y} \geq \mathbf{b}$ then also $\mathbf{A}(\alpha\mathbf{x} + (1 - \alpha\mathbf{y})) \geq \mathbf{b}$. But this follows from basic matrix algebra:

$$\begin{array}{rcl} \mathbf{A}(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) & = & \alpha\mathbf{A}\mathbf{x} + (1 - \alpha)\mathbf{A}\mathbf{y} \\ & \geq & \alpha\mathbf{b} + (1 - \alpha)\mathbf{b} \\ & = & \mathbf{b}. \end{array}$$

□

Proof of Theorem 5.2.8. This is a proof by contradiction: Suppose there is an optimal point $\mathbf{x}^*$ that is an inner point of the feasible region. Then, for a small enough r, all points that are not further away from $\mathbf{x}^*$ than the distance r belong to the feasible region. In particular, the point

$$\mathbf{w} \quad = \quad \mathbf{x}^* + \frac{r}{2}\frac{\mathbf{c}}{\|\mathbf{c}\|}$$

will belong to the feasible region. Here $\|\mathbf{c}\|$ denotes the Euclidean distance:

$$\|\mathbf{c}\| \quad = \quad \sqrt{\sum_{i=1}^{n} c_i^2},$$

and thus $\mathbf{c}/\|\mathbf{c}\|$ is a unit-length vector pointing at the same direction as $\mathbf{c}$.

Now, at point $\mathbf{w}$ we get for the objective function $f(\mathbf{x}) = \mathbf{c}'\mathbf{x}$ that

$$\mathbf{c}'\mathbf{w} \quad = \quad \mathbf{c}'\mathbf{x}^* + \frac{r}{2}\frac{\mathbf{c}'\mathbf{c}}{\|\mathbf{c}\|} \quad = \quad \mathbf{c}'\mathbf{x}^* + \frac{r}{2}\|\mathbf{c}\| \quad > \quad \mathbf{c}'\mathbf{x}^*, \tag{5.4.1}$$

since

$$\mathbf{c}'\mathbf{c} \quad = \quad \|\mathbf{c}\|^2.$$

But inequality (5.4.1) is a contradiction, since $\mathbf{x}^*$ was optimal. So the assumption that $\mathbf{x}^*$ was an inner point must be wrong. □

Proof of Theorem 5.2.9. This proof requires rather deep knowledge of linear algebra, and of linear spaces, although the idea itself is not so complicated if you can visualize n-dimensional spaces. (Taking $n = 3$ should give you the idea.)

Let $\mathbf{x}^*$ be an optimal solution, and let $z^* = \mathbf{c}'\mathbf{x}^*$ be the optimal value. We already know, by Theorem 5.2.8, that $\mathbf{x}^*$ is in the boundary of the feasible region. So, at least one constraints is active. Let now V be the subspace of $\mathbb{R}^n$ spanned by the active constraints at point $\mathbf{x}^*$. Let k be the dimension of V. If $k = n$, then $\mathbf{x}^*$ is a boundary point, and we are done. Suppose then that $k < n$. Then V is a proper subspace or $\mathbb{R}^n$ and any vector in $\mathbb{R}^n$ can be written as an orthogonal sum of a vector from the subspace V and a vector from the orthogonal complement $V^\perp$. Let us write the vector $\mathbf{c}$ this way: $\mathbf{c} = \mathbf{c}_V + \mathbf{c}_{V^\perp}$.

Next we show that $\mathbf{c}$ belongs to the subspace V, i.e., $\mathbf{c}_{V^\perp} = \mathbf{0}$. Suppose the contrary: $\mathbf{c}_{V^\perp} \neq \mathbf{0}$. This means that there is a small $\epsilon > 0$ such that $\mathbf{x}^+ = \mathbf{x}^* + \epsilon\mathbf{c}_{V^\perp}$ is a feasible solution. But now

$$\begin{aligned} z^+ &= \mathbf{c}'\mathbf{x}^+ \\ &= \mathbf{c}'\mathbf{x}^* + \epsilon\mathbf{c}'\mathbf{c}_{V^\perp} \\ &= z^* + \epsilon\mathbf{c}_V'\mathbf{c}_{V^\perp} + \epsilon\mathbf{c}_{V^\perp}'\mathbf{c}_{V^\perp} \\ &= z^* + \epsilon\|\mathbf{c}_{V^\perp}\|^2 \\ &> z^*, \end{aligned}$$

which is a contradiction, since z^* was the optimal value.

Since $k < n$ there is a non-zero point $\mathbf{w}$ in $V^\perp$ such that $\tilde{\mathbf{x}} = \mathbf{x}^* + \alpha\mathbf{w}$ is feasible when $\alpha > 0$ is small enought, and not feasible when $\alpha > 0$ is too large. Now, let α be just small enough for $\mathbf{x}$ to be feasible. Then at point $\tilde{\mathbf{x}}$ at least one more constraint will become active. So, the space $\tilde{V}$ associated to the point $\tilde{\mathbf{x}}$ has at least the dimension $k+1$. Moreover, the point $\tilde{\mathbf{x}}$ is at least as good as the point $\mathbf{x}^*$, since

$$\begin{aligned} \tilde{z} &= \mathbf{c}'\tilde{\mathbf{x}} \\ &= \mathbf{c}'(\mathbf{x}^* + \alpha\mathbf{w}) \\ &= z^* + \alpha\mathbf{c}'\mathbf{w} \\ &= z^*. \end{aligned}$$

(Here we used the fact that $\mathbf{c}$ belongs to V.)

Now, $\tilde{\mathbf{x}}$ is "closer to a corner" than $\mathbf{x}^*$, since it has $k+1$ active constraints. By taking $\tilde{\mathbf{x}}$ to be the new $\mathbf{x}^*$ and repeating the procedure described above $n-k-1$ times we will find an optimal solution in a corner. □

Proof of Theorem 5.3.1. Let $\mathbf{y}$ be some BFS of the LP. Theorem 5.3.1 is proved if we can show that $\mathbf{c}'\mathbf{y} \leq \mathbf{c}'\mathbf{x}$.

Now, since $\mathbf{y}$ is feasible there is $\mathbf{t} \geq \mathbf{0}$ such that $\mathbf{A}\mathbf{y} + \mathbf{t} = \mathbf{b}$. Denote $\mathbf{w} = \mathbf{x} - \mathbf{y}$. Then $\mathbf{A}\mathbf{w} = \mathbf{s} - \mathbf{t}$, and

$$\begin{aligned} \mathbf{c}\mathbf{y} &= \mathbf{c}'(\mathbf{x} + \mathbf{w}) \\ &= \mathbf{c}'\mathbf{x} + \mathbf{c}'\mathbf{w} \\ &= \mathbf{c}'\mathbf{x} + \mathbf{v}'\mathbf{A}\mathbf{w} - \mathbf{u}'\mathbf{w} \\ &= \mathbf{c}'\mathbf{x} + \mathbf{v}'\mathbf{s} - \mathbf{v}'\mathbf{t} - \mathbf{u}'\mathbf{w} \\ &= \mathbf{c}'\mathbf{x} + \mathbf{v}'\mathbf{s} - \mathbf{v}'\mathbf{t} - \mathbf{u}'\mathbf{y} + \mathbf{u}'\mathbf{x} \\ &= \mathbf{c}'\mathbf{x} - \mathbf{v}'\mathbf{t} - \mathbf{u}'\mathbf{y} \\ &\leq \mathbf{c}'\mathbf{x}. \end{aligned}$$

So, $\mathbf{x}$ was indeed optimal. □

Chapter 6

Simplex Method

This chapter is the very hard core of this course!

Here we learn how to solve LPs manually. You may think that it is useless to know such arcane things: We should use computers, you might say — especially since the practical LPs are so large that no-one really solves them manually. This criticism is valid. But, we do not study how to solve LPs manually in order to solve them manually in practical problems (although it is not a *completely* useless skill). We study how to solve them manually in order *to understand* them!

This lecture is adapted from [2, Ch. 2] and [4, Ch. 4].

6.1 Towards Simplex Algorithm

Checking Corners

Theorem 5.2.9 told us that the optimal solution of an LP is in one of the corners of the feasible region. So, it seems that we have a very simple algorithm for finding the optimum: Just check all the corners! And, indeed, this naïve approach works well with such petty examples that we have in this course. The problem with this naïve approach in practice is the so-called combinatorial curse, a.k.a. the curse of dimensionality: An LP with n decision variables and m constraints has

$$\binom{n}{m} = \frac{n!}{(n-m)!m!}$$

corners.

Let us consider the curse of dimensionality more closely: Consider an LP with 30 decision variables and 15 constraints. This LP has

$$\binom{30}{15} = 155{,}117{,}520$$

corners. Suppose you have a computer that checks 100 corners per second (this is pretty fast for today's computers, and right-out impossible if you program

with Java™). Then it would take almost three weeks for the computer to check all the 155,117,520 corners. You may think this is not a problem: Maybe three weeks is not such a long time, and a problem with 30 decision variables is way bigger than anything you would encounter in the real life anyway. Well, think again! Three weeks is a long time if you need to update your optimal solution in a changing environment, say, daily, and LPs with at 30 decision variables are actually rather small. Indeed, let us be a bit more realistic now: Consider a shop owner who has 200 different products in her stock (a rather small shop). Suppose the shop owner has 100 constraints (not unreasonable) and a supercomputer that checks 100 million corners per second (very optimistic, even if one does not program with Java™). Then checking all the corners to optimize the stock would take 6.89×10^{44} years. The author doubts that even the universe can wait that long!

The bottom line is that *checking all the corners will take too much time* even with a fast computer and a good programmer.

Simplex Idea

The general idea of the Simplex algorithm is that you do not check *all* the corners. The following list explains the Simplex algorithm in a meta-level. We call the steps Meta-Steps since they are in such a general level that they are not immediately useful. In the same way the three Meta-Step algorithm could be called a Meta-Simplex algorithm.

We shall see later how the Meta-Steps can be implemented in practice.

Meta-Step 1 Start with some corner.

Meta-Step 2 Check if the corner is optimal. If so, you have found the optimum, and the algorithm terminates. Otherwise go to the next Meta-Step.

Meta-Step 3 Move to an adjacent corner. Of all the adjacent corners choose the best one. Go back to Meta-Step 2.

One hopes that in moving around the corners one hits the optimal corner pretty soon so that one does not have to check all the corners.

To use the meta-algorithm described above we have to:

- identify the corners analytically,
- know how to tell if a chosen corner is optimal,
- know how to go to the best adjacent corner.

Once the points raised above are solved we have a genuine algorithm. This algorithm is given in the next section. Before that we have to discuss how to prepare an LP before it can be used in the Simplex algorithm.

Slack Forms

Before we can use the Simplex algorithm we must transform the LP into a so-called canonical slack form.

We start with the slack form. Here is an informal definition of the slack form: An LP is in slack form, if

1. It is a maximization problem.
2. The constraints are equalities, rather than inequalities.
3. The Right Hand Side (RHS) of each constraint is non-negative.

6.1.1 Example.

$$\begin{array}{lrcrcrl} \max z & = & 4x_1 & + & 8x_2 & & (0) \\ \text{s.t.} & & x_1 & + & 2x_2 & \leq\ 500 & (1) \\ & & x_1 & + & x_2 & \geq\ 100 & (2) \\ & & & & x_1, x_2 & \geq\ 0 & (3) \end{array}$$

Let us transform the LP in Example 6.1.1 above into a slack form.

This is a maximization problem already, so we do not have to touch the line (0).

Line (1) is an inequality. We can transform it into an equality by adding an auxiliary non-negative *slack* (or surplus) variable s_1: We obtain the constraint

$$x_1 + 2x_2 + s_1 = 500 \qquad (1')$$

and, since we assumed that the slack s_1 was non-negative, we have the sign constraints

$$x_1, x_2, s_1 \geq 0 \qquad (3')$$

The interpretation of the slack variable s_1 is that it tells how much of the resource (1) is unused.

Let us then consider line (2). We see that the LP is not in standard form. We could change it into a standard form by multiplying the inequality (2) by -1. But that would make the RHS of (2) negative, which is not good. Instead we ad — or actually subtract — an auxiliary non-negative *excess* variable e_2 to the inequality. We obtain the equality

$$x_1 + x_2 - e_2 = 100 \qquad (2')$$

and the sign constraints

$$x_1, x_2, s_1, e_2 \geq 0 \qquad (3'').$$

The interpretation of the excess is opposite to that of the slack: Excess e_2 tells how much the minimal requirement (2) is, well, excessed.

Now, the LP in 6.1.1 is transformed into a slack form:

$$\begin{array}{lrcrcrcrl}
\max z & = & 4x_1 & + & 8x_2 & & & & & (0) \\
\text{s.t.} & & x_1 & + & 2x_2 & + & s_1 & = & 500 & (1') \\
& & x_1 & + & x_2 & & - \; e_2 & = & 100 & (2') \\
& & & & & & x_1, x_2, s_1, e_2 & \geq & 0 & (3'')
\end{array}$$

Solving this slack form with decisions x_1, x_2, s_1, e_2 is equivalent to solving the original LP with decisions x_1, x_2.

Here is the formal definition of the slack form:

6.1.2 Definition. An LP is in *slack form* if it is of the type

$$\begin{array}{lrcl}
\max z = \mathbf{c}'\mathbf{x} & & & \\
\text{s.t.} & [\mathbf{A}\ \mathbf{S}] \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} & = & \mathbf{b} \\
& \mathbf{x}, \mathbf{s} & \geq & \mathbf{0}
\end{array}$$

where $\mathbf{b} \geq \mathbf{0}$. Here $\mathbf{s}$ is the vector of slacks/excesses and $\mathbf{S}$ is the diagonal matrix containing the coefficients of the slacks and the excesses: $+1$ for slack and -1 for excess.

Here is an algorithm for transforming a standard form LP

$$\begin{array}{lrcl}
\max z & = & \mathbf{c}'\mathbf{x} & \\
\text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\
& \mathbf{x} & \geq & \mathbf{0}
\end{array}$$

into a slack form:

Step 1: Add slacks If the b_i in the constraint i is non-negative add a slack (or surplus):

$$\begin{array}{rcl}
a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n & \leq & b_i \\
\rightsquigarrow \quad a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + s_i & = & b_i.
\end{array}$$

Step 2: Add excesses If the b_i in the constraint i is negative change the direction of the inequality (thus making the RHS $-b_i$ non-negative), and add an excess:

$$\begin{array}{rcl}
a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n & \leq & b_i \\
\rightsquigarrow \quad -a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{in}x_n - e_i & = & -b_i.
\end{array}$$

Steps 1 and 2 must be done to each constraint.

6.1.3 Remark. The index i of the slack/excess refers to resources. For example, if $s_3 = 2$ it means that 2 units of the resource 3 is unused.

6.1.4 Remark. We have two kinds of auxiliary variables: slacks (or surpluses) and excesses. Mathematically there is no need to differentiate between them: Excess is just negative slack (or negative surplus). Indeed, in some textbooks slacks are used for both the surpluses and the excesses. However, making the sign difference makes the problem, and the solution, easier to interpret, especially since typically the all the coefficients of an LP are non-negative.

Finally, let us give the definition of the canonical slack form.

6.1.5 Definition. A slack form LP is *canonical slack form* if each constraint equation has a unique variable with coefficient 1 that does not appear in any other constraint equation.

6.1.6 Remark. Note that the slack form we constructed in Example 6.1.1 is not a canonical one. This is basically due to "wrong sign" -1 of the excess variable. Indeed, if there were a slack instead of an excess in the constraint (2) we would have a canonical form: The slacks would be the unique variables with coefficient 1 that do not appear in any other constraint equation. We shall see in Chapter 7 how to transform the slack form of 6.1.1 into a canonical form by using the Big M method. In this lecture we shall have to confine ourselves to more simple problems.

Basic Feasible Solutions, Basic Variables, and Non-Basic Variables

There is still one more concept — or two, or three, depending on how you count — that we have to discuss before we can present the Simplex algorithm: That of Basic Variables (BV) and Non-Basic Variables (NBV).

Basic variables (BV) and non-basic variables (NBV) are related to the corners, or the basic feasible solutions (BFS), of an underdetermined linear system. So, what we are discussing in this subsection is related to the Meta-Step 1 of the Meta-Simplex algorithm.

Before going into formal definitions let us consider the following problem:

6.1.7 Example. Leather Ltd. manufactures two types of belts: the regular model and the deluxe model. Each type requires 1 unit of leather. A regular belt requires 1 hour of skilled labor and deluxe belt requires 2 hours of of skilled labor. Each week 40 units of leather and 60 hours of skilled labor are available. Each regular belt contributes €3 to profit, and each deluxe belt contributes €4 to profit.

Leather Ltd. wants to maximize its profit.

Let us build the LP for Leather Ltd.

First, we have to choose the decision variables. So, what is there for Leather Ltd. to decide? The number of products to produce! So, Leather Ltd. has the following decision variables:

$$\begin{aligned} x_1 &= \text{number of regular belts manufactured} \\ x_2 &= \text{number of deluxe belts manufactured} \end{aligned}$$

Second, we have to find the objective function. What is it that Leather Ltd. wants to optimize? The profit! What is the Leather Ltd.'s profit? Well, each regular belt contributes €3 to the profit, and each deluxe belt contributes €4 to the profit. Since we denoted the number of regular belts produced by x_1 and the number of deluxe belts produced by x_2 the profit to be maximized is

$$z = 3x_1 + 4x_2.$$

Finally, we have to find the constraints. So, what are the restrictions Leather Ltd. has to satisfy in making the belts? There are two restrictions: available labor and available leather. Let us consider the leather restriction first. There are only 40 units of leather available, and producing one regular belt requires 1 unit of leather. So does producing one deluxe belt. So, the leather constraint is

$$x_1 + x_2 \leq 40.$$

Let us then consider the labor constraint. There are only 60 hours of labor available. Each regular belt produced consumes 1 hour of labor and each deluxe belt produced consumes 2 hours of labor. So, the labor constraint is

$$x_1 + 2x_2 \leq 60.$$

Putting what we have just obtained together we obtain the LP for Leather Ltd. Here it is:

$$\begin{array}{lrcrcrcr} \max z & = & 3x_1 & + & 4x_2 & & & \\ \text{s.t.} & & x_1 & + & x_2 & \leq & 40 \\ & & x_1 & + & 2x_2 & \leq & 60 \\ & & & & x_1, x_2 & \geq & 0 \end{array}$$

Following the algorithm given after Definition 6.1.2 we can transform the LP above into a slack form. Here is what we get:

$$\begin{array}{lrcrcrcrcr} \max z & = & 3x_1 & + & 4x_2 & & & & \\ \text{s.t.} & & x_1 & + & x_2 & + & s_1 & = & 40 \\ & & x_1 & + & 2x_2 & & + s_2 & = & 60 \\ & & & & & & x_1, x_2, s_1, s_2 & \geq & 0 \end{array}$$

Let us then solve this slack form by using the method of Brutus Forcius (108–44 BC), which corresponds to checking all the corners. The Brutus's method is based on the following observation, listed here as Remark 6.1.8:

6.1.8 Remark. Consider the constraints of an LP in slack form. This is a linear system with m equations and $n+m$ unknowns: n actual decision variables and m slacks. Since $n+m > m$ this linear system is underdetermined. In principle, to solve a system of m equations requires only m variables. The remaining n variables can be set to zero.

So, according to Remark 6.1.8, we choose successively $2 = m$ of the $4 = n+m$ variables x_1, x_2, s_1, s_2 to be our basic variables (BV) and set the remaining $2 = n$ variables to be zero (NBV) and solve the constraint system. If the solution turns out to be feasible (it may not be since we are omitting the non-negativity constraints here) we check the value of the objective at this solution. Since we this way check all the BFSs of the system we must find the optimal value.

The next table lists the results:

BVs	Linear system	x_1	x_2	s_1	s_2	BFS	z	Pt
s_1, s_2	$0 + 0 + s_1 = 40$ $0 + 0 + s_2 = 60$	0	0	40	60	Yes	0	F
x_2, s_2	$0 + x_2 + 0 = 40$ $0 + 2x_2 + s_2 = 60$	0	40	0	20	Yes	120	B
x_2, s_1	$0 + x_2 + s_1 = 40$ $0 + 2x_2 + 0 = 60$	0	60	−20	0	No	–	D
x_1, s_2	$x_1 + 0 + 0 = 40$ $x_1 + 0 + s_2 = 60$	40	0	0	−20	No	–	A
x_1, s_1	$x_1 + 0 + s_1 = 40$ $x_1 + 0 + 0 = 60$	30	0	10	0	Yes	120	C
x_1, x_2	$x_1 + x_2 + 0 = 40$ $x_1 + 2x_2 + 0 = 60$	20	20	0	0	Yes	140	E

From this table we read that the decision

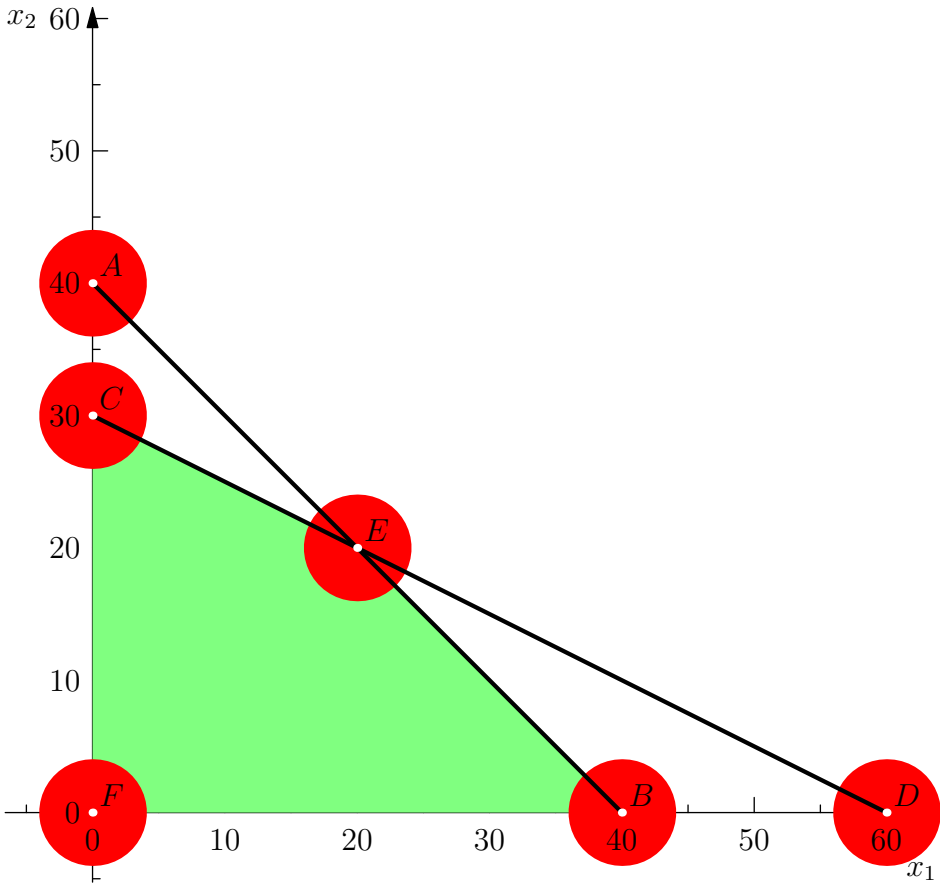
$$x_1 = 20, \quad x_2 = 20, \quad s_1 = 0, \quad s_2 = 0$$

is optimal. The corresponding optimal value is

$$z \;=\; €140.$$

So, we have solved Leather Ltd.'s problem. Note that both of the slacks, s_1 and s_2, are NBV, i.e. zeros, at the optimal decision. This means that at the optimal solution the resources, leather and skilled labor, are fully utilized. This full utilization of the resources is not uncommon in LPs, but it is not always the case. Sometimes it may be optimal not to use all your resources.

Next picture illustrates the situation. The centers of the red balls are the candidate BFSs (Pts in the previous table). Note that only the points B, C, E, and F are actual BFSs. The optimal BFS is the point E.



6.2 Simplex Algorithm

Simplex Steps

Step 1: Transform the LP into canonical slack form Transforming LP into a slack form has been explained in the previous section. For now, let us just hope that the said slack form is also a canonical one. It will be if there are no excesses. If there are excesses then the slack form most likely will not be canonical — unless you are extremely lucky. From the canonical slack form we construct the *first Simplex Tableau*. The first Simplex tableau is the canonical slack form where

- The 0th row represents the objective function as a 0th constraint as
$$z - \mathbf{c}'\mathbf{x} \quad = \quad 0.$$
- The variables that have unique row with 1 as coefficient, and 0 as coefficient in all other rows, will be chosen to be the BVs. Typically, the slacks are chosen to be the BVs. In that case the decisions are

set to be zero, and thus the first Simplex tableau will be solved for the slacks.

So, most typically, the slack form LP

$$\begin{array}{lrcccccl}
\max z & = & c_1x_1 & +\cdots+ & c_nx_n & & & \\
\text{s.t.} & & a_{11}x_1 & +\cdots+ & a_{1n}x_n & +s_1 & & = b_1 \\
& & a_{21}x_1 & +\cdots+ & a_{2n}x_n & +s_2 & & = b_2 \\
& & & \vdots & & & & \vdots \\
& & a_{m1}x_1 & +\cdots+ & a_{mn}x_n & \cdots & +s_m & = b_m \\
& & & & & x_1,\ldots,x_n,s_1,\ldots,s_m & & \geq 0
\end{array}$$

becomes

$$\begin{array}{lrcccccl}
\max z & & & & & & & \\
\text{s.t.} & z & -c_1x_1 & -\cdots- & c_nx_n & & & = 0 \\
& & a_{11}x_1 & +\cdots+ & a_{1n}x_n & +s_1 & & = b_1 \\
& & a_{21}x_1 & +\cdots+ & a_{2n}x_n & +s_2 & & = b_2 \\
& & & \vdots & & & & \vdots \\
& & a_{m1}x_1 & +\cdots+ & a_{mn}x_n & \cdots & +s_m & = b_m \\
& & & & & x_1,\ldots,x_n,s_1,\ldots,s_m & & \geq 0
\end{array}$$

Since we have to keep track of the BVs, this form is then represented as the Simplex tableau

Row	z	x_1	$\cdots$	x_n	s_1	s_2	$\cdots$	s_m	BV	RHS
0	1	$-c_1$	$\cdots$	$-c_n$	0	0	$\cdots$	0	$z=$	0
1	0	a_{11}	$\cdots$	a_{1n}	1	0	$\cdots$	0	$s_1=$	b_1
2	0	a_{21}	$\cdots$	a_{2n}	0	1	$\cdots$	0	$s_2=$	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
m	0	a_{m1}	$\cdots$	a_{mn}	0	0	$\cdots$	1	$s_m=$	b_m

From this tableau one readily reads the BFS related to the BVs $s_1,\ldots,s_m$: $[s_1\ \cdots\ s_m]' = [b_1\ \cdots\ b_m]'$; and $[x_1\ \cdots\ x_n]' = [0\ \cdots\ 0]'$.

Step 2: Check if the current BFS is optimal In the first Simplex tableau the BVs are $s_1,\ldots,s_m$, and a BFS related to this solution is $x_1=0,\ldots,x_n=0$, $s_1=b_1,\ldots,s_m=b_m$. The value of the objective can be read from the 0th row:

Row	z	x_1	$\cdots$	x_n	s_1	s_2	$\cdots$	s_m	BV	RHS
0	1	$-c_1$	$\cdots$	$-c_n$	0	0	$\cdots$	0	$z=$	0

This solution is hardly optimal. Indeed, suppose that the coefficients c_i are non-negative (as is usually the case). But now all the decisions x_i related to the coefficients c_i are zero, as they are NBVs. But then,

obviously increasing the value of any x_i will increase the value of the objective z.

Let us then consider the general case. Suppose that, after some steps, we have come up with a Simplex tableau with the 0th row

Row	z	x_1	$\cdots$	x_n	s_1	s_2	$\cdots$	s_m	BV	RHS
0	1	d_1	$\cdots$	d_n	d_{n+1}	d_{n+2}	$\cdots$	d_{n+m}	$z =$	z^*

where all the coefficients d_i are non-negative for all the NBVs. Then making any NBV a BV would decrease the value of the objective. So, the criterion for the optimality is: **The Simplex tableau is optimal, if in the 0th row there are no negative coefficients in any NBVs.**

If the tableau is optimal the algorithm terminates, and the optimal value and decision can be read from the BV and RHS columns.

Step 3: Determine the entering variable If the BFS is not optimal, we have to change the BVs. One of the NBVs will become a BV (entering), and one of the old BVs will become a NBV (leaving). **The entering variable will be the one with smallest coefficient in the 0th row.** Indeed, this way we increase the value of the objective z the most.

Step 4: Determine the leaving variable In Step 3 we chose some variable to enter as a new BV. Now we have to make one of the old BVs to leave to be a NBV. Now each BV in a Simplex tableau is associated to some row. The **leaving BV will the one associated to the row that wins the ratio test** (the smallest value is the winner)

$$\frac{\text{RHS of row}}{\text{Coefficient of entering varaible in row}}.$$

The idea of the ratio test is, that we shall increase the entering variable as much as possible. At some point the increasing of the entering variable will force one of the BVs to become zero. This BV will then leave. The ratio test picks up the row associated to the leaving variable.

Step 5: Find a new BFS Now we have a new system of BVs. Next we have to solve the Simplex tableau in terms of the new BVs. This can be done by using the Gauss–Jordan method. Then we have a new Simplex tableau, and we **go back to Step 2**.

6.2.1 Remark. The Step 1 above corresponds to the Meta-Step 1. The Step 2 corresponds to the Meta-Step 2. The Steps 3–5 correspond to the Meta-Step 3.

Dakota Furniture's Problem

6.2.2 Example. The Dakota Furniture Company manufactures desks, tables, and chairs. The manufacture of each type of furniture requires lumber and two types of skilled labor: finishing labor and carpentry labor. The amount of each resource needed to make each type of furniture is given in the table below:

Resource	**Desk**	**Table**	**Chair**
Lumber	8 units	6 units	1 unit
Finishing hours	4 hours	2 hours	1.5 hours
Carpentry hours	2 hours	1.5 hours	0.5 hours

At present, 48 units of lumber, 20 finishing hours, and 8 carpentry hours are available. A desk sells for €60, a table for €30, and a chair for €20. Dakota believes that demand for desks and chairs is unlimited, but at most 5 tables can be sold.

Since the available resources have already been purchased, Dakota wants to maximize total revenue.

As a modelling problem Dakota's problem is very similar to Giapetto's problem 3.3.1. After making some comparisons on how we modelled Giapetto's problem we notice that we should define the decision variables as

$$\begin{aligned} x_1 &= \text{number of desks produced} \\ x_2 &= \text{number of tables produced} \\ x_3 &= \text{number of chairs produced} \end{aligned}$$

and that Dakota should solve the following LP:

$$\begin{array}{lrcrcrcr} \max z & = & 60x_1 & + & 30x_2 & + & 20x_3 & \\ \text{s.t.} & & 8x_1 & + & 6x_2 & + & x_3 & \leq & 48 \\ & & 4x_1 & + & 2x_2 & + & 1.5x_3 & \leq & 20 \\ & & 2x_1 & + & 1.5x_2 & + & 0.5x_3 & \leq & 8 \\ & & & & x_2 & & & \leq & 5 \\ & & & & & & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Dakota Furniture's Solution with Simplex

Step 1: We start by transforming the Dakota's LP into a slack form. Since all the inequalities are of type $\leq$ we have no excesses, and consequently we obtain the canonical slack form

$$\begin{array}{lrcrcrcrcrcrcr}
\max z = & 60x_1 & + & 30x_2 & + & 20x_3 & & & & & & & & \\
\text{s.t.} & 8x_1 & + & 6x_2 & + & x_3 & + & s_1 & & & & & = & 48 \\
& 4x_1 & + & 2x_2 & + & 1.5x_3 & & & + & s_2 & & & = & 20 \\
& 2x_1 & + & 1.5x_2 & + & 0.5x_3 & & & & & + & s_3 & = & 8 \\
& & & x_2 & & & & & & & & & = & 5 \\
& & & & & & & & & x_1, x_2, x_3, s_1, s_2, s_3, s_4 & & & \geq & 0
\end{array}$$

Taking s_1, s_2, s_3, s_4 to be our first BVs our first Simplex tableau for Dakota is

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	-60	-30	-20	0	0	0	0	$z =$	0
1	0	8	6	1	1	0	0	0	$s_1 =$	48
2	0	4	2	1.5	0	1	0	0	$s_2 =$	20
3	0	2	1.5	0.5	0	0	1	0	$s_3 =$	8
4	0	0	1	0	0	0	0	1	$s_4 =$	5

Step 2: We check if the current Simplex tableau is optimal. The 0th row is now

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	-60	-30	-20	0	0	0	0	$z =$	0

We see that there is a NBV x_1 with negative coefficient -60. So the first Simplex tableau is not optimal. (Well, one does not expect to get an optimal solution by slacking off!)

Step 3: We determine the entering variable. Since x_1 has the smallest coefficient in row 0, increasing x_1 will allow the objective z to increase most. So, x_1 will enter as a new BV.

Step 4: We determine the leaving variable. The ratio test gives us

$$\begin{array}{lcll}
\text{Row 1 limit in on } x_1 & = & 48/8 & = 6 \\
\text{Row 2 limit in on } x_1 & = & 20/4 & = 5 \\
\text{Row 3 limit in on } x_1 & = & 8/2 & = 4 \\
\text{Row 4 limit in on } x_1 & = & \text{No limit, since } x_i\text{'s coefficient is non-positive} &
\end{array}$$

So, Row 3 wins the ratio test. Since s_3 the the BV associated to row 3, s_3 is no longer a BV.

Step 5: Now we have new BVs: s_1, s_2, x_1, s_4 (remember x_1 replaced s_3). This means we have the unsolved Simplex tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	-60	-30	-20	0	0	0	0	$z =$	0
1	0	8	6	1	1	0	0	0	$s_1 =$	48
2	0	4	2	1.5	0	1	0	0	$s_2 =$	20
3	0	2	1.5	0.5	0	0	1	0	$x_1 =$	8
4	0	0	1	0	0	0	0	1	$s_4 =$	5

Now we have to solve this Simplex tableau in terms of the BVs. This means that each row must have coefficient 1 for its BV, and that BV must have coefficient 0 on the other rows. This can be done with EROs in the following way:

ERO1: We create a coefficient of 1 for x_1 in row 3 by multiplying row 3 by 0.5. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	-60	-30	-20	0	0	0	0	$z =$	0
1	0	8	6	1	1	0	0	0	$s_1 =$	48
2	0	4	2	1.5	0	1	0	0	$s_2 =$	20
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a 0 coefficient for x_1 in row 0, we replace the row 0 with $60(\text{row } 3) + \text{row } 0$. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	15	-5	0	0	30	0	$z =$	240
1	0	8	6	1	1	0	0	0	$s_1 =$	48
2	0	4	2	1.5	0	1	0	0	$s_2 =$	20
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a 0 coefficient for x_1 in row 1, we replace row 1 with $-8(\text{row3}) + \text{row } 1$. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	15	-5	0	0	30	0	$z =$	240
1	0	0	0	-1	1	0	-4	0	$s_1 =$	16
2	0	4	2	1.5	0	1	0	0	$s_2 =$	20
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a 0 coefficient for x_1 in row 2, we replace row 2 with $-4(\text{row } 3) + \text{row } 2$. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	15	−5	0	0	30	0	$z =$	240
1	0	0	0	−1	1	0	−4	0	$s_1 =$	16
2	0	0	−1	0.5	0	1	−2	0	$s_2 =$	4
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

Now we see that this Simplex tableau is solved: Each of the BVs have coefficient 1 on their own rows and coefficient 0 in other rows. So, we go now back to Step 2.

Step 2: We check if the Simplex tableau above is optimal. It is not, since the NBV x_3 has negative coefficient on row 0.

Step 3: We determine the entering variable. In this case it is obvious: x_3 enters.

Step 4: We determine the leaving variable. The ratio test gives us

$$\begin{array}{lcrcr} \text{Row 1 limit in on } x_3 & = & \text{No limit} & & \\ \text{Row 2 limit in on } x_3 & = & 4/0.5 & = & 8 \\ \text{Row 3 limit in on } x_3 & = & 4/0.25 & = & 16 \\ \text{Row 4 limit in on } x_3 & = & \text{No limit} & & \end{array}$$

So, row 2 wins the ratio test. Since s_2 was the BV of row 2, s_2 will leave and become a NBV.

Step 5: Now we have new BVs: s_1, x_3, x_1, s_4, since s_2 was replaced with x_3 in the previous step. So, we have the following unsolved Simplex tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	15	−5	0	0	30	0	$z =$	240
1	0	0	0	−1	1	0	−4	0	$s_1 =$	16
2	0	0	−1	0.5	0	1	−2	0	$x_3 =$	4
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

To solve this tableau we must invoke the Gauss–Jordan method again:

ERO1: To create a coefficient of 1 for x_3 in row 2, we multiply the row 2 by 2. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	15	−5	0	0	30	0	$z =$	240
1	0	0	0	−1	1	0	−4	0	$s_1 =$	16
2	0	0	−2	1	0	2	−4	0	$x_3 =$	8
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a coefficient 0 for x_3 in row 0, we replace row 0 with 5(row 2) + row 0. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	5	0	0	10	10	0	$z =$	280
1	0	0	0	−1	1	0	−4	0	$s_1 =$	16
2	0	0	−2	1	0	2	−4	0	$x_3 =$	8
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a coefficient 0 for x_3 in row 1, we replace row 1 with row 2 + row 1. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	5	0	0	10	10	0	$z =$	280
1	0	0	−2	0	1	2	−8	0	$s_1 =$	24
2	0	0	−2	1	0	2	−4	0	$x_3 =$	8
3	0	1	0.75	0.25	0	0	0.5	0	$x_1 =$	4
4	0	0	1	0	0	0	0	1	$s_4 =$	5

ERO2: To create a coefficient 0 for x_3 in row 3, we replace row 3 with −0.25(row 3) + row 3. Now we have the tableau

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	5	0	0	10	10	0	$z =$	280
1	0	0	−2	0	1	2	−8	0	$s_1 =$	24
2	0	0	−2	1	0	2	−4	0	$x_3 =$	8
3	0	1	1.25	0	0	−0.5	1.5	0	$x_1 =$	2
4	0	0	1	0	0	0	0	1	$s_4 =$	5

Now we see that this Simplex tableau is solved: Each of the BVs have coefficient 1 on their own rows and coefficient 0 in other rows. So, we go now back to Step 2.

Step 2: We check if the Simplex tableau is optimal. We see that it is! Indeed, all the NBVs x_2, s_2, s_3 have non-negative coefficients in row 0.

Finally, let us interpret the result: The number of desks, tables, and chairs Dakota Furniture should manufacture is 2, 0, and 8. With this decision Dakota's revenue is €280. Of the resources: 24 units of lumber is left unused: $s_1 = 24$. All the other actual resources are fully used: $s_2 = 0, s_3 = 0$, but the market demand for tables is not used at all $s_5 = 5$, since no tables are manufactured.

Dakota Furniture's Solution with `glpsol`

We show briefly how to solve the Dakota's problem of Example 6.2.2 with `glpsol`.

Here are the contents of the file `dakota.mod`, where the Dakota's problem is described in GNU MathProg modelling language. There are no slacks here in the code: GLPK will do the transformations for you internally.

```
#
# Dakota's problem
#
# This finds the optimal solution for maximizing Dakota's revenue
#

/* Decision variables */
var x1 >=0;  /* desk */
var x2 >=0;  /* table */
var x3 >=0;  /* chair */

/* Objective function */
maximize z: 60*x1 + 30*x2 + 20*x3;

/* Constraints */
s.t.     Lumber : 8*x1 +   6*x2 +     x3 <= 48;
s.t.  Finishing : 4*x1 +   2*x2 + 1.5*x3 <= 20;
s.t.  Carpentry : 2*x1 + 1.5*x2 + 0.5*x3 <=  8;
s.t.     Demand :            x2          <= 40;

end;
```

So, issue the command

```
glpsol -m dakota.mod -o dakota.sol
```

Now, you should get in your console something like the following:

```
Reading model section from dakota.mod...
21 lines were read
Generating z...
Generating Lumber...
Generating Finishing...
Generating Carpentry...
Generating Demand...
Model has been successfully generated
glp_simplex: original LP has 5 rows, 3 columns, 13 non-zeros
glp_simplex: presolved LP has 3 rows, 3 columns, 9 non-zeros
lpx_adv_basis: size of triangular part = 3
*     0:   objval =   0.000000000e+00   infeas =   0.000000000e+00 (0)
*     2:   objval =   2.800000000e+02   infeas =   0.000000000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (114563 bytes)
lpx_print_sol: writing LP problem solution to 'dakota.sol'...
```

The file `dakota.sol` should now contain the following report:

```
Problem:    dakota
Rows:       5
Columns:    3
Non-zeros:  13
Status:     OPTIMAL
Objective:  z = 280 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            280
     2 Lumber       B             24                          48
     3 Finishing    NU            20                          20            10
     4 Carpentry    NU             8                           8            10
     5 Demand       B              0                          40

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B              2             0
     2 x2           NL             0             0                          -5
     3 x3           B              8             0

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err. = 7.11e-15 on row 1
        max.rel.err. = 7.11e-17 on row 2
        High quality

KKT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err. = 3.55e-15 on column 2
        max.rel.err. = 9.87e-17 on column 2
        High quality

KKT.DB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

End of output
```

You should compare this output with the last, optimal, Simplex tableau. Indeed, recall that the optimal Simplex tableau was

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	s_4	BV	RHS
0	1	0	5	0	0	10	10	0	$z =$	280
1	0	0	-2	0	1	2	-8	0	$s_1 =$	24
2	0	0	-2	1	0	2	-4	0	$x_3 =$	8
3	0	1	1.25	0	0	-0.5	1.5	0	$x_1 =$	2
4	0	0	1	0	0	0	0	1	$s_4 =$	5

where

$$
\begin{aligned}
x_1 &= \text{Number of desks produced} \\
x_2 &= \text{Number of tables produced} \\
x_3 &= \text{Number of chairs produced} \\
s_1 &= \text{Amount of lumber unused} \\
s_2 &= \text{Number of finishing hours unused} \\
s_3 &= \text{Number of carpentry hours unused} \\
s_4 &= \text{Demand for tables unused}
\end{aligned}
$$

Now, compare this to the following part of the `glpsol`'s report

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B            280
     2 Lumber       B             24                          48
     3 Finishing    NU            20                          20            10
     4 Carpentry    NU             8                           8            10
     5 Demand       B              0                          40

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B              2             0
     2 x2           NL             0             0                          -5
     3 x3           B              8             0
```

We will talk more about the connection later in Chapter 8 where we talk about sensitivity analysis. Now we just make some remarks. First, note the connections

$$
\begin{aligned}
x_1 &= \texttt{x1} \\
x_2 &= \texttt{x2} \\
x_3 &= \texttt{x3} \\
s_1 &= \texttt{Lumber} \text{ (unused)} \\
s_2 &= \texttt{Finishing} \text{ (unused)} \\
s_3 &= \texttt{Carpentry} \text{ (unused)} \\
s_4 &= \texttt{Demand} \text{ (unused)}
\end{aligned}
$$

Whenever a variable is BV in the optimal Simplex tableau `glpsol` reports its status in the `St` column as bounded by the symbol `B`. So, you see that the status for `x1`, `x3`, `z`, `Lumber`, and `Demand` are bounded, since they are BVs. The NBVs `glpsol` will mark as `NU` or `NL`. The `Marginal` column in `glpsol`'s report is related to the 0th row in the `glpsol`'s report. The marginals, or the shadow prices, are very important in sensitivity analysis. They tell how much the value of the optimal solution will change if the corresponding constraint is relaxed

by one unit. (The negative marginal with x_2 is a slightly different story, since x_2 is not a constraint, but a decision variable.)

Chapter 7

More on Simplex Method

This chapter is adapted from [2, Ch. 2] and [4, Ch. 4].

7.1 Big M Algorithm

Problem with Canonical Slack Form

Recall that the Simplex algorithm requires a starting BFS, i.e., the first Simplex tableau must be solved with some BVs. So far we have not had any excesses, and consequently the first Simplex tableau was solved with the slack variables as BVs. If an LP has inequalities of type $\geq$ with non-negative RHS, or equalities, then finding the starting BFS can be difficult, or even right out impossible. The following example illustrates this point.

7.1.1 Example. Bevco manufactures an orange-flavored soft drink called Oranj by combining orange soda and orange juice. Each unit of orange soda contains 0.5 units of sugar and 1 mg of vitamin C. Each unit of orange juice contains 0.25 units of sugar and 3 mg of vitamin C. It costs Bevco €0.02 to produce a unit of orange soda and €0.03 to produce a unit of orange juice. Bevco's marketing department has decided that each each bottle of Oranj must be of size 10 units, and must contain at least 20 mg of vitamin C and at most 4 units of sugar.

How can the marketing department's requirements be met at minimum cost?

Let us construct the LP for Bevco.

We have had many examples already that we have modelled as LPs. Remember, e.g., Giapetto 3.3.1, Leather Ltd. 6.1.7, and Dakota 6.2.2. In all those problems we had to decide how many of each products we should produce in

order to maximize the profit. Bevco's problem is different. For starters, there is only one product.

So, what is the decision Bevco must make? The only thing that is actually not fixed in Example 7.1.1, and thus open for decision, is the actual mixture of orange soda and orange juice in a bottle of Oranj. So, the decision variables are:

$$\begin{aligned} x_1 &= \text{Number of units of orange soda in a bottle of Oranj} \\ x_2 &= \text{Number of units of orange juice in a bottle of Oranj} \end{aligned}$$

What about the objective then? What does Bevco want to maximize or minimize? We see that Bevco wants to minimize the cost of producing Oranj. So this is a minimization problem. So, what is the cost of producing one bottle of Oranj? Well, recall that it costs €0.02 to produce one unit of orange soda, and €0.03 to produce one unit of orange juice. So, the objective function to be minimized is

$$z = 2x_1 + 3x_2$$

(here we measure in Cents rather than in Euros in order to avoid fractions).

So far we have found out the decision variables and the objective. What about the constraints then?

Remember the marketing department's demands: Each bottle of Oranj must contain at least 20 mg of vitamin C and at most 4 units of sugar.

Let us consider first the sugar demand. Each unit of orange soda has 0.5 units of sugar in it and each unit of orange juice has 0.25 units of sugar in it. Since a bottle of Oranj must not contain more than 4 units of sugar we obtain the constraint

$$0.5x_1 + 0.25x_2 \leq 4$$

(here the fractions cannot be avoided — at least not easily).

The vitamin C constraint is similar to the sugar constraint, but opposite. Indeed, each unit of orange soda contains 1 mg of vitamin C, and each unit of orange juice contains 3 mg of vitamin C. Since there must be at least 20 mg of vitamin C in a bottle of Oranj, we obtain

$$x_1 + 3x_2 \geq 20.$$

There is still one more constraint. This constraint is not so obvious as the sugar and vitamin C constraints, but it must be included. Recall that each bottle of Oranj must be of size 10 units. So, as the bottle of Oranj only contains orange soda and orange juice, we must have

$$x_1 + x_2 = 10.$$

Finally, note the classical sign constraints

$$x_1, x_2 \quad \geq \quad 0.$$

Indeed, it would be pretty hard to put negative amount of either orange soda or orange juice in a bottle.

We have found the LP for Bevco:

$$\begin{array}{lrcrcrl}
\min z & = & 2x_1 & + & 3x_2 & & \\
\text{s.t.} & & 0.5x_1 & + & 0.25x_2 & \leq & 4 \qquad \text{(sugar constraint)} \\
& & x_1 & + & 3x_2 & \geq & 20 \qquad \text{(vitamin C constraint)} \\
& & x_1 & + & x_2 & = & 10 \qquad \text{(10 units in a bottle of Oranj)} \\
& & & & x_1, x_2 & \geq & 0
\end{array}$$

Let us then try to apply the Simplex method to this LP. First we turn the LP into a slack form (note that this is a minimization problem). We obtain

$$\begin{array}{lrcrcrcrcrcll}
\max & -z & & & & & & & & & & & \\
\text{s.t.} & -z & + & 2x_1 & + & 3x_2 & & & & = & 0 & & (0) \\
& & & 0.5x_1 & + & 0.25x_2 & + & s_1 & & = & 4 & & (1) \\
& & & x_1 & + & 3x_2 & & & - \; e_2 & = & 20 & & (2) \\
& & & x_1 & + & x_2 & & & & = & 10 & & (3) \\
& & & & & & & x_1, x_2, s_1, e_2 & & \geq & 0 & & (4)
\end{array} \tag{7.1.2}$$

The problem now is that the slack form above is not in canonical form. We have three genuine constraint equations (1)–(3), but there are no three variables in the set x_1, x_2, s_1, e_2 that could be taken as BVs under which the constraint system above could be solved *while keeping the RHSs still non-negative*.

7.1.3 Remark. There are two reasons why Bevco's slack form turned out to be non-canonical. One is the vitamin C constraint

$$x_1 + 3x_2 \quad \geq \quad 20.$$

This is a lower-bound constraint that will give us an excess variable — and excess variables have "wrong" signs. The other is the equality constraint

$$x_1 + x_2 \quad = \quad 10.$$

This constraint does not give us any slacks or excesses. So, we fall short of variables.

Solution with Artificial Variables

The problem with the system (1)–(3) is that we do not have enough variables. So, the solution is obvious: We introduce new *artificial variables* where needed.

Now, row (1) in (7.1.2) is fine: It has s_1. Rows (2) and (3) of (7.1.2), on the other hand, are not fine: They lack variables with coefficient 1 that do not appear in any other row. So, we introduce artificial variables: a_2 will enter row (2) and a_3 will enter row (3). We get the system

$$
\begin{array}{llrcrcrcrcrcrcrcrcl}
 & \max & -z & & & & & & & & & & & & & & \\
 & \text{s.t.} & -z & + & 2x_1 & + & 3x_2 & & & & & & & = & 0 & & (0) \\
 & & & & 0.5x_1 & + & 0.25x_2 & + & s_1 & & & & & = & 4 & & (1) \\
(7.1.4) & & & & x_1 & + & 3x_2 & & & - & e_2 & + & a_2 & = & 20 & & (2) \\
 & & & & x_1 & + & x_2 & & & & & & a_3 & = & 10 & & (3) \\
 & & & & & & & & & & x_1, x_2, s_1, e_2, a_2, a_3 & & & \geq & 0 & & (4)
\end{array}
$$

Now we have a BFS. Indeed, taking s_1, a_2, a_3 to be the BVs we obtain

$$z = 0, \quad s_1 = 4, \quad a_2 = 20, \quad a_3 = 10.$$

But now we have a small problem. What would guarantee that an optimal solution to (7.1.4) is also an optimal solution to (7.1.2)? Well, actually, nothing! Indeed, we might find an optimal solution to (7.1.4) where some of the artificial variables are strictly positive. In such case it may turn out that the corresponding solution to (7.1.2) is not even feasible. For example, in (7.1.4) it can be shown that the solution

$$z = 0, \quad s_1 = 4, \quad a_2 = 20, \quad a_3 = 10, \quad x_1 = 0, \quad x_2 = 0$$

is optimal. But this solution is not feasible for the original problem. Indeed, this solution contains no vitamin C, and puts 0 units of soda and juice in a bottle. So, this solution cannot possibly be optimal in the original problem, as it is not even a solution.

The critical point is: In the optimal solution all the artificial variables must be zero. How is this achieved? By changing the objective function! Recall that the original objective function for Bevco was (in max form)

$$\max \ -z \ = \ -2x_1 - 3x_2.$$

Now, let M be a very very very very very very very very very very very very big number — if you approve that $0 \times \infty = 0$, you may think that $M = +\infty$. Consider then the objective function

$$\max \ -z \ = \ -2x_1 - 3x_2 \ - Ma_2 - Ma_3.$$

Now allowing a_2 or a_3 be strictly positive should penalize the value of $-z$ so much that the solution could not possibly be optimal. This means that an

optimal solution to the system,

(7.1.5)

$$
\begin{array}{lrcrcrcrcrcrcl}
\max & -z & & & & & & & & & & & & \\
\text{s.t.} & -z & + & 2x_1 & + & 3x_2 & & & +Ma_2 & +Ma_3 & = & 0 & & (0)\\
 & & & 0.5x_1 & + & 0.25x_2 & +s_1 & & & & = & 4 & & (1)\\
 & & & x_1 & + & 3x_2 & & -e_2 & +a_2 & & = & 20 & & (2)\\
 & & & x_1 & + & x_2 & & & & +a_3 & = & 10 & & (3)\\
 & & & & & & & x_1, x_2, s_1, e_2, a_2, a_3 & & & \geq & 0 & & (4)
\end{array}
$$

should have $a_2 = 0$ and $a_3 = 0$. But then an optimal solution of (7.1.5) is also an optimal solution to the original problem (7.1.2) of Example 7.1.1.

The system (7.1.5) is not yet in canonical slack form. There is one more trick left. To solve (7.1.5) in terms of the prospective BVs s_1, a_2, a_3 we must remove a_2 and a_3 from row 0. This is done by using the ERO2 (two times): Replace row 0 with row $0 - M(\text{row } 2) - M(\text{row } 3)$. This way we obtain the system

(7.1.6)

$$
\begin{array}{lrcrcrcrcrcrcl}
\max & -z & & & & & & & & & & & & \\
\text{s.t.} & -z & + & (2-2M)x_1 & + & (3-4M)x_2 & & & +Me_2 & & = & -30M & & (0)\\
 & & & 0.5x_1 & + & 0.25x_2 & +s_1 & & & & = & 4 & & (1)\\
 & & & x_1 & + & 3x_2 & & -e_2 & +a_2 & & = & 20 & & (2)\\
 & & & x_1 & + & x_2 & & & & +a_3 & = & 10 & & (3)\\
 & & & & & & & x_1, x_2, s_1, e_2, a_2, a_3 & & & \geq & 0 & & (4)
\end{array}
$$

This is a canonical slack form. The BVs are s_1, a_2, a_3.

Now we have a canonical slack form (7.1.6). So, we can carry out the Simplex algorithm. The next steps are the Simplex algorithm steps.

Step 1: Our first Simplex — or Simplex/Big M — tableau is the system (7.1.6):

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$2-2M$	$3-4M$	0	M	0	0	$-z =$	$-30M$
1	0	0.5	0.25	1	0	0	0	$s_1 =$	4
2	0	1	3	0	-1	1	0	$a_2 =$	20
3	0	1	1	0	0	0	1	$a_3 =$	10

Step 2: We check for optimality. Our Simplex tableau is not optimal since there are negative coefficients in row 0 for the NBVs x_1 and x_2 (remember that M is a very very very very big number).

Step 3: We determine the entering variable. Now, when M is big enough — and M is always big enough — we have that

$$3 - 4M \quad \leq \quad 2 - 2M.$$

So, x_2 will enter as a new BV.

Step 4: We determine the leaving variable. The ratio tests give us

$$\begin{array}{lcrcl} \text{Row 1 limit in on } x_2 & = & 4/0.25 & = & 16 \\ \text{Row 2 limit in on } x_2 & = & 20/3 & = & 6.667 \\ \text{Row 3 limit in on } x_2 & = & 10/1 & = & 10 \end{array}$$

So, Row 2 wins the ratio test. Since a_2 the the BV associated to row 2, a_2 is no longer a BV.

Step 5: Now we have new BVs: s_1, x_2, a_3 (remember x_2 replaced a_2). This means we have the unsolved Simplex tableau

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$2-2M$	$3-4M$	0	M	0	0	$-z=$	$-30M$
1	0	0.5	0.25	1	0	0	0	$s_1=$	4
2	0	1	3	0	-1	1	0	$x_2=$	20
3	0	1	1	0	0	0	1	$a_3=$	10

We have to solve this tableau in terms of the BVs s_1, x_2, a_3 by using the Gauss–Jordan method. Applying the Gauss–Jordan method here is a bit tricky since we have the symbol M. So, we cannot just count with numbers — we have to do some algebra. Let us start. First we eliminate x_2 from row 0. As a first step to that direction we use ERO1 and multiply row 2 by 1/3. We obtain the tableau

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$2-2M$	$3-4M$	0	M	0	0	$-z=$	$-30M$
1	0	0.5	0.25	1	0	0	0	$s_1=$	4
2	0	$\frac{1}{3}$	1	0	$\frac{-1}{3}$	$\frac{1}{3}$	0	$x_2=$	$\frac{20}{3}$
3	0	1	1	0	0	0	1	$a_3=$	10

Next, we eliminate x_2 from row 0. This is done by adding $(4M-3)$row 2 to to row 0. Our new Simplex tableau is then

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$\frac{3-2M}{3}$	0	0	$\frac{3-M}{3}$	$\frac{4M-3}{3}$	0	$-z=$	$\frac{-60-10M}{3}$
1	0	0.5	0.25	1	0	0	0	$s_1=$	4
2	0	$\frac{1}{3}$	1	0	$\frac{-1}{3}$	$\frac{1}{3}$	0	$x_2=$	$\frac{20}{3}$
3	0	1	1	0	0	0	1	$a_3=$	10

Next two steps are to eliminate x_1 from the rows 1 and 3. We omit the details, and just state the solved Simplex tableau:

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$\frac{3-2M}{3}$	0	0	$\frac{3-M}{3}$	$\frac{4M-3}{3}$	0	$-z=$	$\frac{-60-10M}{3}$
1	0	$\frac{5}{12}$	0	1	$\frac{1}{12}$	$\frac{-1}{12}$	0	$s_1=$	$\frac{7}{3}$
2	0	$\frac{1}{3}$	1	0	$\frac{-1}{3}$	$\frac{1}{3}$	0	$x_2=$	$\frac{20}{3}$
3	0	$\frac{2}{3}$	0	0	$\frac{1}{3}$	$\frac{-1}{3}$	1	$a_3=$	$\frac{10}{3}$

Step 2: We check for optimality. The tableau above is not optimal: The NBV x_1 has negative coefficient.

Step 3: WE determine the entering variable. The NBV x_1 has the smallest coefficient among all NBVs, so it enters.

Step 4: We determine the leaving variable. The ratio test gives us

$$\begin{aligned} \text{Row 1 limit in on } x_1 &= \tfrac{7}{3}\Big/\tfrac{5}{12} = 5.6 \\ \text{Row 2 limit in on } x_1 &= \tfrac{20}{3}\Big/\tfrac{1}{3} = 20 \\ \text{Row 3 limit in on } x_1 &= \tfrac{10}{3}\Big/\tfrac{2}{3} = 5 \end{aligned}$$

So, Row 3 wins the ratio test. Since a_3 the BV associated to row 2, a_3 is no longer a BV.

Step 5: Now we have new BVs: s_1, x_2, x_1, and a new Simplex tableau we have to solve with the Gauss–Jordan method. We omit the cumbersome details. Here is the solved tableau:

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	0	0	0	$\frac{1}{2}$	$\frac{2M-1}{2}$	$\frac{2M-3}{2}$	$-z =$	-25
1	0	0	0	1	$\frac{-1}{8}$	$\frac{1}{8}$	$\frac{-5}{8}$	$s_1 =$	$\frac{1}{4}$
2	0	0	1	0	$\frac{-1}{2}$	$\frac{1}{2}$	$\frac{-1}{2}$	$x_2 =$	5
3	0	1	0	0	$\frac{1}{2}$	$\frac{-1}{2}$	$\frac{3}{2}$	$x_1 =$	5

Step 2: We check for optimality. We see that the tableau is optimal! Indeed, there are no strictly negative coefficients in the 0th row for NBVs.

So, the solution for Bevco is to put equal amount — 5 units and 5 units — of orange soda and orange juice in a bottle of Oranj. Then the production cost of a bottle is minimized, and it is €0.25 (remember we counted in Cents). At the optimal solution $s_1 = 0.25$, which means that there is only 3.75 units sugar in a bottle of Oranj (remember that the maximal allowed sugar content was 4 units). The excess variable $e_2 = 0$. This means that there is exactly 20 mg of vitamin C in the bottle of Oranj. Finally, note that the artificial variables a_2 and a_3 are both zero, as they should be.

7.1.7 Remark. It may turn out that in solving the Big M analog of an LP the optimal solution has non-zero artificial variables. If this happens, it means that the original LP does not have any feasible solutions.

Big M Steps

Let us formalize, or algorihmize, the Big M method we explained by an example in the previous subsection.

Step 1: Start with a slack form The slack form is constructed in the same way as in the plain Simplex case. Remember that the RHSs must be non-negative, and that the problem must be a maximization problem.

Step 2: Add artificial variables To each constraint row, say i, that does not have a slack variable s_i, add an artificial variable a_i. For each artificial variable a_i subtract the value Ma_i from the objective function z. This means, for each of the artificial variables a_i, adding the value Ma_i in the 0th row in the column corresponding the BV a_i.

Step 3: Construct the first Simplex tableau Solve the system you got in Step 2 in terms of the slacks and the artificial variables. This is done by subtracting from the 0th row M times each row that has an artificial variable. Now, the slacks and the artificial variables will be the BVs, and the system is solved in terms of the BVs. Otherwise the first Simplex tableau is the same as in the plain Simplex case.

Step 4: Carry out the Simplex algorithm In Step 3 we constructed the first Simplex tableau. Now this tableau is solved for the slacks and artificial variables, and we may start the Simplex algorithm. Find the optimal BFS of the system by using the Simplex algorithm.

Step 5: Check feasibility Check that in the optimal solution obtained by the Simplex algorithm the artificial variables are zero. If this is the case we have a solution. If at least one of the artificial variables is strictly positive the original LP does not have feasible solutions.

7.2 Simplex Algorithm with Non-Unique Optima

All the LP examples we have had so far had a unique optimal point (probably, I haven't really checked that properly). In this subsection we discuss what happens with the Simplex — or the Simplex/Big M — algorithm if there are many optima, unbounded optimum, or no optima at all.

In this section we shall give all the examples simply as LPs, without any associated story.

Many Bounded Optima

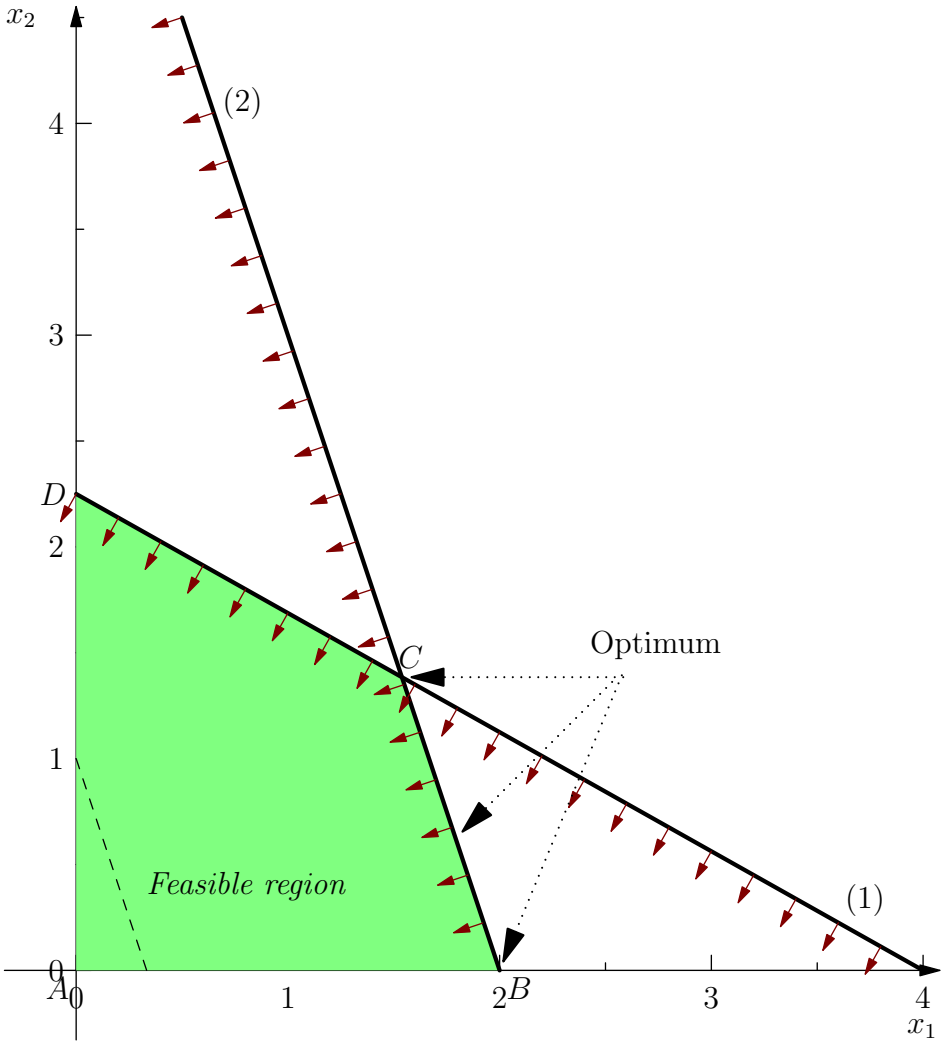
7.2.1 Example. Consider the LP

$$\begin{array}{lrcrcll} \min z & = & -6x_1 & - & 2x_2 & & \\ \text{s.t.} & & 2x_1 & + & 4x_2 & \leq 9 & (1) \\ & & 3x_1 & + & x_2 & \leq 6 & (2) \\ & & & & x_1, x_2 & \geq 0 & (3) \end{array}$$

This is a minimization problem. So, before we do anything else, we transform it into a maximization problem. Here is the transformed problem:

$$\begin{array}{lrcrcll} \max z & = & 6x_1 & + & 2x_2 & & \\ \text{s.t.} & & 2x_1 & + & 4x_2 & \leq 9 & (1) \\ & & 3x_1 & + & x_2 & \leq 6 & (2) \\ & & & & x_1, x_2 & \geq 0 & (3) \end{array} \tag{7.2.2}$$

Here is the graphical representation of the LP of Example 7.2.1 (or actually, of the transformed problem (7.2.2) where we have switched z to $-z$). The isoprofit lines are dashed.



From the picture it is clear that the are many optimal points. Indeed, all the points in the line segment from B to C are optimal — and only the points in the line segment from B to C are optimal. Let us see now what the Simplex algorithm says about this.

First, we transform the LP of Example 7.2.1 — actually the LP (7.2.2) — into a slack form. We obtain

$$\begin{array}{lrcrcrcrcl}
\max & z & & & & & & & & \\
\text{s.t.} & z & - & 6x_1 & - & 2x_2 & & & = & 0 \\
 & & & 2x_1 & + & 4x_2 & + & s_1 & = & 9 \\
 & & & 3x_1 & + & x_2 & & + & s_2 & = & 6 \\
 & & & & & & & x_1, x_2, s_1, s_2 & \geq & 0
\end{array}$$

We see that this slack form is canonical, and we get the first Simplex tableau without resorting to the Big M method:

Row	z	x_1	x_2	s_1	s_2	BV	RHS
0	1	−6	−2	0	0	$z =$	0
1	0	2	4	1	0	$s_1 =$	9
2	0	3	1	0	1	$s_2 =$	6

From the tableau we read that x_1 should enter and s_2 should leave. Solving, by using the Gauss–Jordan, the resulting tableau in terms of the BVs s_1, x_2 gives us the tableau

Row	z	x_1	x_2	s_1	s_2	BV	RHS
0	1	0	0	0	2	$z =$	12
1	0	0	3.333	1	−0.667	$s_1 =$	5
2	0	1	0.333	0	0.333	$x_1 =$	2

We see that the tableau is optimal. The BFS found corresponds to the point B in the previous picture. So, we have found the optimal solution $x_1 = 2$, $x_2 = 0$, and $z = 12$. So, everything is fine. Except that we know — from the picture — that the found optimum is not unique. There are others. How does the Simplex tableau tell us this? Well, there is a NBV decision variable x_2 with coefficient 0 in the 0th row. This means that making the decision variable x_2 to enter as BV would not change the value of the objective. So let us — just for fun — make x_2 to enter, and s_1 leave. We get, after Gauss–Jordan, the following tableau

Row	z	x_1	x_2	s_1	s_2	BV	RHS
0	1	0	0	0	2	$z =$	12
1	0	0	1	0.3	−0.2	$x_2 =$	1.5
2	0	1	0	−0.1	0.4	$x_1 =$	1.5

We see that we have a new optimum. This time $x_1 = 1.5$ and $x_2 = 1.5$. This optimum corresponds to the point C in the previous picture. All the other optima are convex combinations of the optima we have just found, i.e. they are in the line segment from B to C.

The bottom line is:

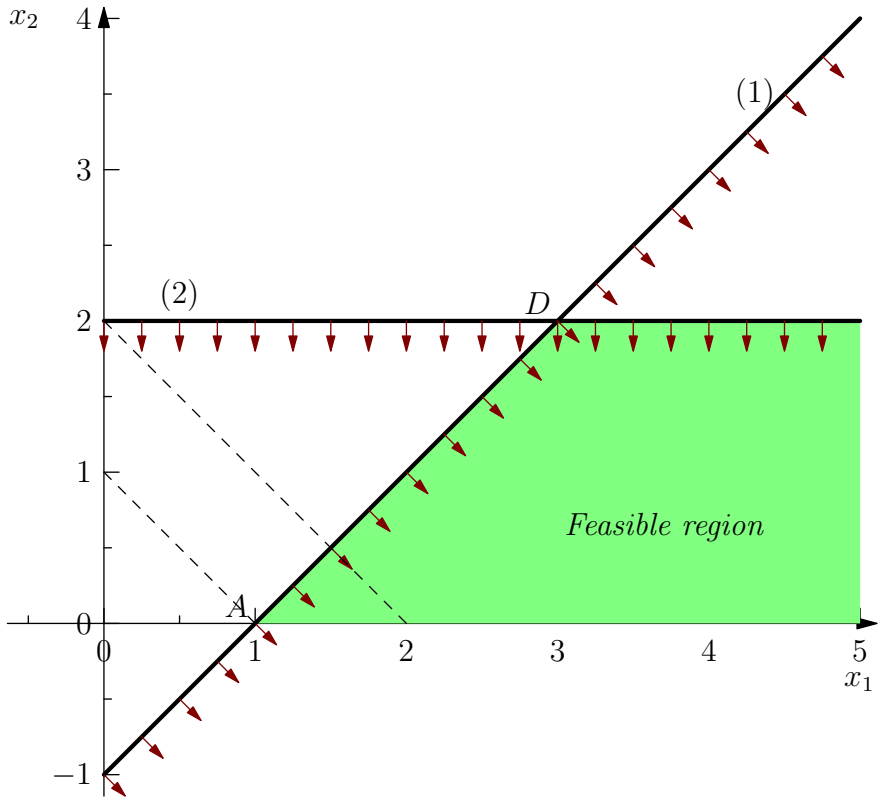
Whenever there is a NBV with coefficient 0 *on the* 0*th row of an optimal tableau, there are many optima.*

Unbounded Optimum

7.2.3 Example. Consider the LP

$$\begin{array}{lrcrcll} \max z & = & x_1 & + & x_2 & & \\ \text{s.t.} & & x_1 & - & x_2 & \geq & 1 \qquad (1) \\ & & & & 6x_2 & \geq & 2 \qquad (2) \\ & & & & x_1, x_2 & \geq & 0 \qquad (3) \end{array}$$

Here is the graphical representation of the LP of Example 7.2.1. The isoprofit lines are dashed.



From the picture we see that this LP has unbounded optimum. Indeed, since the feasible region continues in the right up to infinity, one finds better and better solutions as one moves the isoprofit line further away from the origin (remember, no matter how far the isoprofit line is from the origin it will cross the x_1-axis somewhere).

Let us then see what the Simplex algorithm have to say about unbounded optimum.

First, we transform the LP of Exercise 7.2.3 into a slack form. We obtain

$$
\begin{array}{llrcrcrcrcl}
 & \max & z & & & & & & & & \\
 & \text{s.t.} & z & - & x_1 & - & x_2 & & & = & 0 \\
(7.2.4) & & & & x_1 & - & x_2 & - & e_1 & = & 1 \\
 & & & & & & 6x_2 & & + \ s_2 & = & 2 \\
 & & & & & & & & x_1, x_2 & \geq & 0
\end{array}
$$

We see that we should use the Big M method here. That would indeed work. We do not go that way, however. Instead, we will be clever this time. We note that the system (7.2.4) is equivalent to the system

$$
\begin{array}{llrcrcrcrcl}
 & \max & z & & & & & & & & \\
 & \text{s.t.} & z & & & - & 2x_2 & - & e_1 & = & 1 \\
 & & & & x_1 & - & x_2 & - & e_1 & = & 1 \\
 & & & & & & 6x_2 & & + \ s_2 & = & 2 \\
 & & & & & & & & x_1, x_2 & \geq & 0
\end{array}
$$

which is a canonical slack form if one chooses x_1 and s_2 as BVs. So, we obtain the first Simplex tableau

Row	z	x_1	x_2	e_1	s_2	BV	RHS
0	1	0	-2	-1	0	$z =$	1
1	0	1	-1	-1	0	$x_1 =$	1
2	0	0	6	0	1	$s_2 =$	2

Now it is obvious that x_2 should enter as BV, and that s_2 should leave. After solving the Tableau with x_1 and x_2 as BVs we obtain the Tableau

Row	z	x_1	x_2	e_1	s_2	BV	RHS
0	1	0	0	-1	2	$z =$	5
1	0	1	0	-1	1	$x_1 =$	3
2	0	0	1	0	1	$x_2 =$	2

Now it is obvious that e_1 should enter as a new BV. So, let us try to determine the leaving variable. But the ratio test will give no limit. This means that no matter how much we increase the value of e_1 the old BVs will remain positive, never reaching the boundary 0 and thus becoming NBVs. So, the conclusion is:

Whenever the ratio test fails to identify a leaving variable the LP in question has an unbounded optimum.

No Optima

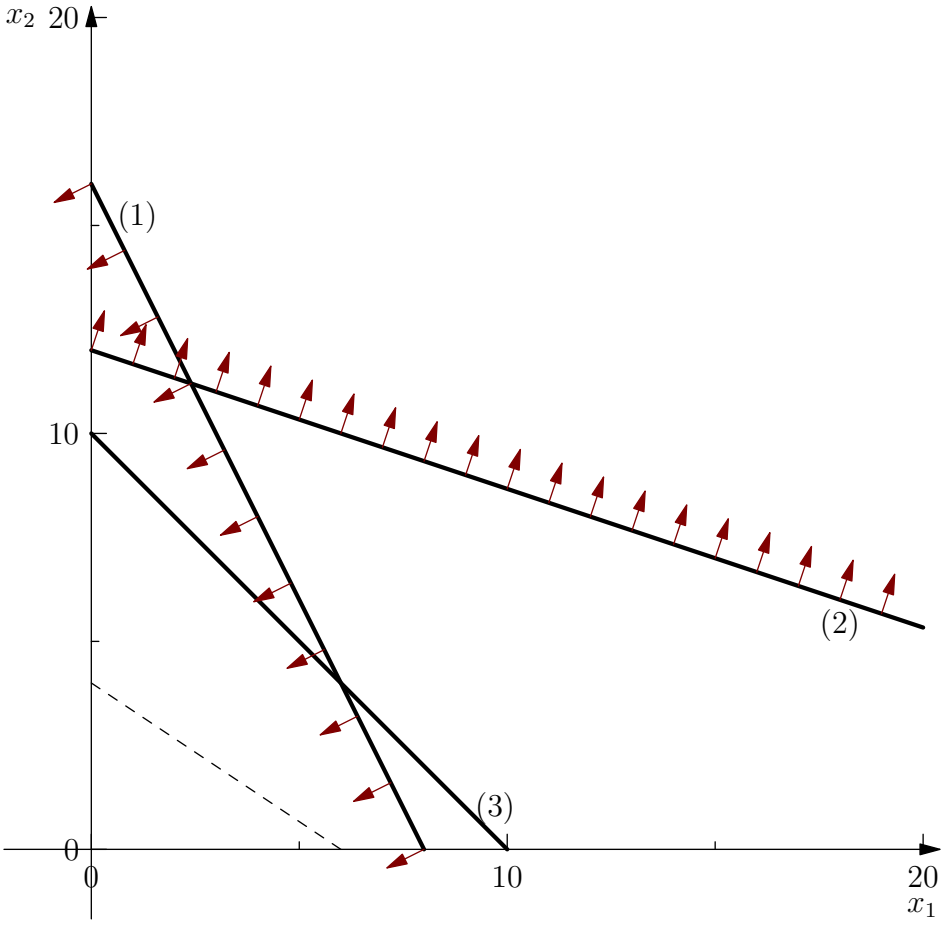
Let us agree that what does not exist is non-unique.

The only way an LP may fail to have an optimal solution is that it has no feasible solutions at all.

7.2.5 Example.

$$
\begin{array}{lrcrcrl}
\min z & = & 2x_1 & + & 3x_2 & & \\
\text{s.t.} & & 0.5x_1 & + & 0.25x_2 & \leq & 4 \qquad (1) \\
& & x_1 & + & 3x_2 & \geq & 36 \qquad (2) \\
& & x_1 & + & x_2 & = & 10 \qquad (3) \\
& & & & x_1, x_2 & \geq & 0 \qquad (4)
\end{array}
$$

Here is the picture that illustrates the LP of Example 7.2.5. The isoprofit line is dashed.



Note that there are no arrows associated to the constraint line (3). This is because (3) is an equality, not an inequality. So, all the feasible solutions of the LP of Example 7.2.5 must be *on* the line (3). So, from the picture it is clear that the constraint (3) makes the feasible region empty, since lines (1)

and (2) don't touch line (3) at the same point. Actually, line (2) does not touch the line (3) at all.

Let us then see what the Simplex algorithm has to say about the LP of Example 7.2.5

To transform this LP into a canonical slack form we need artificial variables. This due to the $\geq$ constraint (2) and the $=$ constraint (3). The first Simplex/Big M tableau for the LP of Example 7.2.5 is

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$2-2M$	$3-4M$	0	M	0	0	$z=$	$-46M$
1	0	0.5	0.25	1	0	0	0	$s_1=$	4
2	0	1	3	0	-1	1	0	$a_2=$	36
3	0	1	1	0	0	0	1	$a_3=$	10

Next we invoke the Simplex/Big M machinery in search for the optimal tableau. We omit the details here. Our final Simplex tableau will be

Row	$-z$	x_1	x_2	s_1	e_2	a_2	a_3	BV	RHS
0	1	$2M-1$	0	0	M	0	$4M-3$	$z=$	$-6M-30$
1	0	0.25	0	1	0	0	-0.25	$s_1=$	1.5
2	0	-2	0	0	-1	1	-3	$a_2=$	6
3	0	1	1	0	0	0	1	$x_2=$	10

We see that this is indeed the final Simplex tableau: All the NBVs have non-negative coefficients in the 0th row. But there is a problem: The artificial variable a_2 is BV and $a_2 = 6 \neq 0$. But the artificial variables should be 0 in the final solution. Indeed, they were penalized heavily by the very very very very big number M in the objective function. Why did not the Simplex algorithm put them to be 0. The reason why the Simplex algorithm failed to put the artificial variables 0 was that otherwise there would not be any solutions. But in the original problem the artificial variables are 0 — or do not exist, which is the same thing. So, the conclusion is:

Whenever there are non-zero artificial variables as BVs in the final Simplex tableau, the original LP does not have any feasible solutions.

7.3 Simplex/Big M Checklist

The following list combines the algorithms presented in this and the previous lectures as checklist on how to solve LPs with Simplex/Big M method. It should be noted that in solving LPs in practice the may be many shortcuts — the checklist presented here is meant to be general rather than efficient.

Step 1: Prepare the first canonical Simplex tableau

- **Step 1-1:** Transform the LP into a standard form.
- **Step 1-2:** Transform the standard form LP into a slack form.
- **Step 1-3:** Transform, if necessary, the slack form into a canonical slack form by adding artificial variables to each constraint row that lacks slacks. After this subtract M times the artificial variables from the objective function, and solve the system — by using the Gauss–Jordan method — in terms of the slacks and the artificial variables.

Step 2: Carry out the Simplex algorithm

- **Step 2-1:** Transform the LP into canonical slack form. (Actually, this is what we just did in Step 1.)
- **Step 2-2:** Check if the current BFS is optimal. There are, omitting the possibility of multiple optimal solutions, three possibilities:
  (a) All the NBVs have non-negative coefficients, and all the artificial variables have zero coefficient: **The algorithm terminates, and an optimum was found**. The optimum can be read from the columns BV and RHS.
  (b) All the NBVs have non-negative coefficients, but some of the artificial variables have non-zero coefficients: **The algorithm terminates, and the LP has no solutions**.
  (c) Some of the NBVs have strictly negative coefficients: The algorithm moves to Step 2-3 (the next step).
- **Step 2-3:** Determine the entering variable. The NBV with smallest coefficient in 0th row will enter.
- **Step 2-4:** Determine the leaving variable. To do this perform the ratio test. Now there are two possibilities
  (a) Some BV wins the ratio test (gets the smallest number). That variable will leave. The algorithm then continues in Step 2-5.
  (b) All the ratios are either negative of $\pm\infty$. In this case **the algorithm terminates, and the LP has an unbounded solution.**
- **Step 2-5:** Find the new BFS for the new BVs by using the Gauss–Jordan method, and **go back to Step 2-2**.

Chapter 8

Sensitivity and Duality

The most important topic of linear programming is of course solving linear programs. We have just covered the topic in the previous lectures. The second-most important topics in linear programming are sensitivity analysis and duality. (In [4, Ch 5] the author claims that sensitivity analysis and duality are the *most* important topics of linear programming. This author disagrees!) This lecture covers – at least the rudiments — of those.

While sensitivity and duality are two distinct topics their connection is so close and profound that the Jane Austen type title "Sensitivity and Duality" is reasonable.

This chapter is adapted from [2, Ch. 2] and [4, Ch. 5].

8.1 Sensitivity Analysis

What and Why is Sensitivity Analysis

When one uses a mathematical model to describe reality one must make approximations. The world is more complicated than the kind of optimization problems that we are able to solve. Indeed, it may well be that the shortest model that explains the universe is the universe itself. Linearity assumptions usually are significant approximations. Another important approximation comes because one cannot be sure of the data one puts into the model. One's knowledge of the relevant technology may be imprecise, forcing one to approximate the parameters $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ in the LP

$$\begin{array}{lrcl} \max & z & = & \mathbf{c}'\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

Moreover, information may change.

Sensitivity Analysis is a systematic study of how, well, sensitive, the solutions of the LP are to *small changes* in the data. The basic idea is to be able to give answers to questions of the form:

1. If the objective function $\mathbf{c}$ changes in its parameter c_i, how does the solution change?
2. If the resources available change, i.e., the constraint vector $\mathbf{b}$ change in its parameter b_i, how does the solution change?
3. If a new constraint is added to the problem, how does the solution change?

We shall give answers to the questions 1 and 2. Question 1 is related to the concept of *reduced cost*, a.k.a. the opportunity cost. Question 2 is related to the concept of *shadow price*, a.k.a. the marginal price. The question 3 will be completely ignored in these lectures.

One approach to these questions is to solve lots and lots of LPs: One LP to each change in the parameters. For example, in Giapetto's problem 3.3.1 there might be uncertainty in what is the actual market demand for soldiers. It was assumed to be 40, but it could be anything between 30 and 50. We could then solve the Giapetto's LP separately for market demands $30, 31, \ldots, 49, 50$. So, we would solve 20 different LPs (21, actually, but who's counting). If it is also assumed that the the profit for soldiers might not be exactly €3 but could be anything between €2.5 and €3.5, then we could also solve the LP separately for profits €2.5, €2.6, ..., €3.4, €3.5. Combining this with the different LPs we got from the uncertainty in the market demand we would then have $20 \times 10 = 200$ different LPs to solve (well, $21 \times 11 = 231$ if you count correctly). This "checking the scenarios" method would work, and it is indeed widely used in practice. This method has only two problems: (1) It is inelegant, and (2) it would involve a large amount of calculations. These problems are, however, not critical. Indeed, solving hundreds of LPs is not that time-consuming with modern computers and efficient algorithms like the Simplex. As for the inelegance of the scenario-based method: Who cares about elegance these days? Nevertheless, we shall try to be at least a little bit elegant in this chapter.

Shadow Prices

There are two central concepts in sensitivity analysis. They are so important that LP solvers will typically print their values in their standard reports. These are the *shadow prices for constraints* and *reduced costs for decision variables*. In this subsection we consider the shadow prices, and show where they are represented in the `glpsol` reports.

8.1.1 Definition. The *Shadow Price* of a constraint is the amount that the objective function value would change if the said constraint is changed by one unit — given that the optimal BVs don't change. The shadow prices are typically denoted as the vector $\pi = [\pi_1 \ \cdots \ \pi_m]'$.

The following remarks of Definition 8.1.1 should help you to understand the concept.

8.1.2 Remark. Note the clause "given that the optimal BVs don't change". This means that the shadow price is valid for small changes in the constraints. If the optimal corner changes when a constraint is changed, then the interpretation of the shadow price is no longer valid. It is valid, however, for all changes that are small enough, i.e., below some critical threshold.

8.1.3 Remark. Shadow prices are sometimes called *Marginal Prices*. E.g., GLPK calls them marginals. This is actually a much more informative name than the nebulous shadow price. Indeed, suppose you have a constraint that limits the amount of labor available to 40 hours per week. Then the shadow price will tell you how much you would be willing to pay for an additional hour of labor. If your shadow price is €10 for the labor constraint, for instance, you should pay no more than €10 an hour for additional labor. Labor costs of less than €10 per hour will increase the objective value; labor costs of more than €10 per hour will decrease the objective value. Labor costs of exactly €10 will cause the objective function value to remain the same.

If you like mathematical formulas — and even if you don't — the shadow prices can be defined as follows: Consider the LP

$$\begin{array}{lrcl} \max & z & = & \mathbf{c}'\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

Now, the optimal solution z^* of this LP is a function of the objective vector $\mathbf{c}$, the technology matrix $\mathbf{A}$, and the constraint vector $\mathbf{b}$. So,

$$z^* \quad = \quad z^*(\mathbf{c}, \mathbf{A}, \mathbf{b}).$$

Then the shadow price π_i associated to the constraint b_i is the partial derivative

$$\pi_i \quad = \quad \frac{\partial z^*}{\partial b_i},$$

or, in vector form,

$$\pi \quad = \quad \frac{\partial z^*}{\partial \mathbf{b}},$$

where

$$\pi = \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_m \end{bmatrix} \quad \text{and} \quad \frac{\partial z^*}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial z^*}{\partial b_1} \\ \vdots \\ \frac{\partial z^*}{\partial b_m} \end{bmatrix}.$$

Suppose now that $\epsilon = [\epsilon_1 \ \cdots \ \epsilon_m]'$ is a small vector, and

$$z^*_\epsilon \quad = \quad z^*(\mathbf{c}, \mathbf{A}, \mathbf{b} + \epsilon)$$

is the optimal value, when the constraints $\mathbf{b}$ are changed by ϵ. Then the first order "Taylor approximation" for the new optimal value is

$$\begin{aligned} z^*_\epsilon &= z^* + \epsilon' \frac{\partial z^*}{\partial \mathbf{b}} \\ &= z^* + \epsilon' \pi, . \end{aligned} \tag{8.1.4}$$

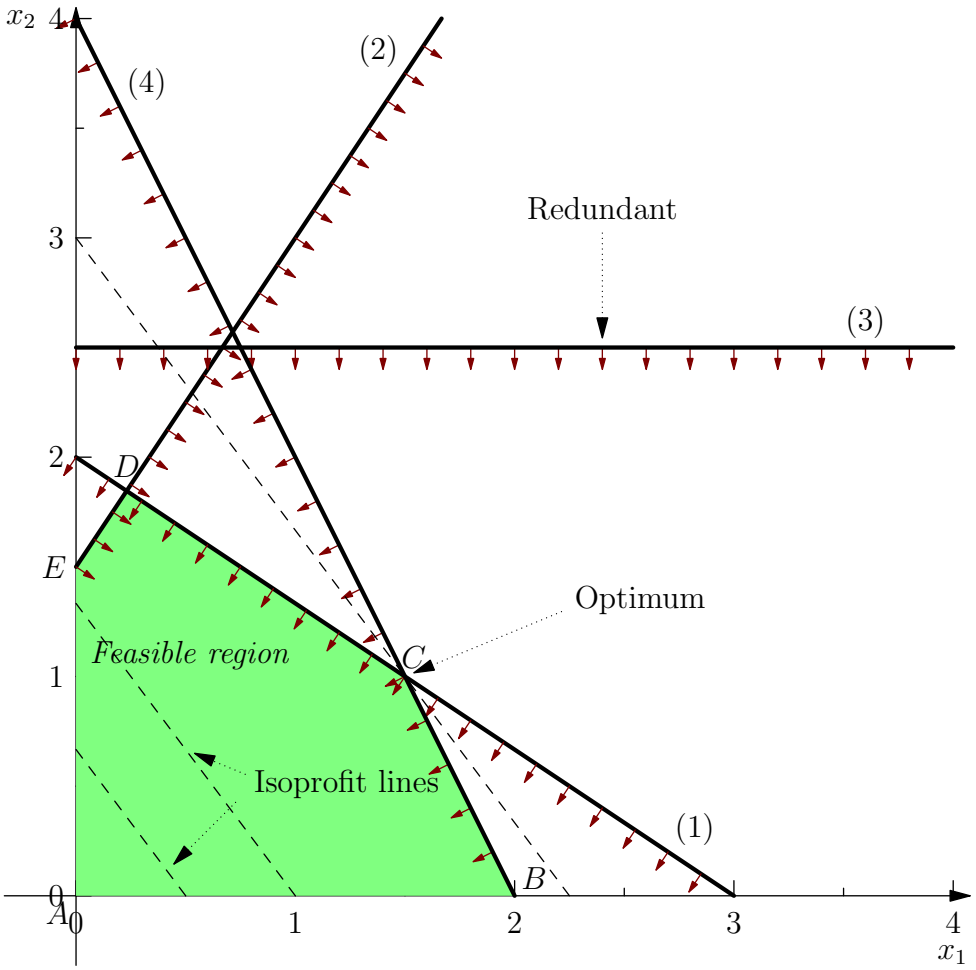
The equality (8.1.4) is valid as long as the elements ϵ_i of ϵ are small enough (in absolute value). If some of the elements of ϵ are too big, then the equality (8.1.4) may fail to be true.

Let us see now how to use formula (8.1.4) in sensitivity analysis.

8.1.5 Example. Consider the LP

$$\begin{array}{lrcrcrcl} \max z & = & 4x_1 & + & 3x_2 & & & (0) \\ \text{s.t.} & & 2x_1 & + & 3x_2 & \leq & 6 & (1) \\ & & -3x_1 & + & 2x_2 & \leq & 3 & (2) \\ & & & & 2x_2 & \leq & 5 & (3) \\ & & 2x_1 & + & x_2 & \leq & 4 & (4) \\ & & & & x_1, x_2 & \geq & 0 & \end{array}$$

Here is the picture representing the LP. You have seen this picture before.



From the picture we read — by moving the isoprofit line away from the origin — that the optimal point for the decision $\mathbf{x} = [x_1 \; x_2]'$ is

$$C \quad = \quad (1.5, 1).$$

Therefore, the optimal value is of the objective is

$$z \quad = \quad 4 \times 1.5 + 3 \times 1 \quad = \quad 9.$$

We also see that the constraints (1) and (4) are active at the optimum. So, changing them should change the optimal value. Indeed, they should have positive shadow prices. In contrast, the constraints (2) and (3) are not active at the optimum. So, changing them — slightly — should have no effect on the optimum. So, both of them should have 0 as their shadow price.

Let us then calculate the shadow prices. We could read the shadow prices from the final Simplex tableau. This would require much work, so we use GLPK instead. Here is the code that solves Example 8.1.5:

```
# Sensitivity analysis for Example 8.1.7
# Part 1 - The original problem

# Decision variables
var x1 >=0;
var x2 >=0;

# Objective
maximize z: 4*x1 + 3*x2;

# Constraints
s.t. r1:  2*x1 + 3*x2 <= 6;
s.t. r2: -3*x1 + 2*x2 <= 3;
s.t. r3:         2*x2 <= 5;
s.t. r4:  2*x1 +   x2 <= 4;

end;
```

And here is the relevant part of the solution:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B              9
     2 r1           NU             6                           6           0.5
     3 r2           B           -2.5                           3
     4 r3           B              2                           5
     5 r4           NU             4                           4           1.5

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B            1.5             0
     2 x2           B              1             0
```

From the report we read that the shadow prices (called `Marginal` in the `glpsol` report) for the constraints (1) (`r1` in the code) and (4) (`r4` in the code) are 0.5 and 1.5, respectively. All other shadow prices are 0 (number omitted in the `glpsol` report). So, the shadow price vector is

$$\pi \quad = \quad \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 1.5 \end{bmatrix}.$$

Let us then try to use formula (8.1.4) to see what happens if the constraints (1)–(4) change. Suppose each constraint is relaxed by 0.5. That means that in formula (8.1.4) we have $\epsilon = [0.5\ 0.5\ 0.5\ 0.5]'$. So, the new optimum should be

$$\begin{aligned} z^*_\epsilon &= z^* + \epsilon'\pi \\ &= 9 + 0.5\times 0.5 + 0.5\times 0 + 0.5\times 0 + 0.5\times 1.5 \\ &= 10. \end{aligned}$$

Is this correct? Let us see. Let us solve the new LP where the constraints are relaxed. So, we have to solve

$$\begin{array}{lrcrcrl}
\max z & = & 4x_1 & + & 3x_2 & & (0)\\
\text{s.t.} & & 2x_1 & + & 3x_2 & \leq 6.5 & (1)\\
& & -3x_1 & + & 2x_2 & \leq 3.5 & (2)\\
& & & & 2x_2 & \leq 5.5 & (3)\\
& & 2x_1 & + & x_2 & \leq 4.5 & (4)\\
& & & & x_1, x_2 & \geq 0 &
\end{array}$$

The GNU MathProg code for this problem is

```
# Sensitivity analysis for Example 8.1.7
# Part 2 - r1,r2,r3,r4 relaxed by 0.5

# Decision variables
var x1 >=0;
var x2 >=0;

# Objective
maximize z: 4*x1 + 3*x2;

# Constraints
s.t. r1:  2*x1 + 3*x2 <= 6.5;
s.t. r2: -3*x1 + 2*x2 <= 3.5;
s.t. r3:         2*x2 <= 5.5;
s.t. r4:  2*x1 +   x2 <= 4.5;

end;
```

The relevant part of the `glpsol` report reads:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B             10
     2 r1           NU           6.5                       6.5           0.5
     3 r2           B          -3.25                      3.5
     4 r3           B              2                      5.5
     5 r4           NU           4.5                       4.5           1.5

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B           1.75             0
     2 x2           B              1             0
```

So, we see that formula (8.1.4) is correct, when $\epsilon \leq [0.5\ 0.5\ 0.5\ 0.5]'$.

Let us then consider a big change. Let $\epsilon = [10\ 10\ 10\ 0]'$. Then formula (8.1.4) would give us

$$\begin{aligned}
z^*_\epsilon &= z^* + \epsilon'\pi\\
&= 9 + 10\times 0.5 + 10\times 0 + 10\times 0 + 0\times 1.5\\
&= 14.
\end{aligned}$$

Let's see what really happens. The MathProg code for this relaxed LP is

```
# Sensitivity analysis for Example 8.1.7
# Part 3 - r1,r2,r3 relaxed by 10; r4 not relaxed

# Decision variables
var x1 >=0;
var x2 >=0;

# Objective
maximize z: 4*x1 + 3*x2;

# Constraints
s.t. r1:  2*x1 + 3*x2 <= 16;
s.t. r2: -3*x1 + 2*x2 <= 13;
s.t. r3:         2*x2 <= 15;
s.t. r4:  2*x1 +   x2 <=  4;

end;
```

And the relevant part of the `glpsol` report tells us that

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B             12
     2 r1           B             12                         16
     3 r2           B              8                         13
     4 r3           B              8                         15
     5 r4           NU             4                          4             3

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           NL             0             0                          -2
     2 x2           B              4             0
```

We see that `glpsol` reports the optimum to be 12. The formula (8.1.4) gave us the optimum 14. So, the change $[10\ 10\ 10\ 0]'$ is not small enough for the formula (8.1.4) to be valid. What actually happened here was that the optimal point jumped corners.

Reduced Costs

Let us then consider the reduced costs. Remember that the shadow prices were associated to the constraints, or — if you like Simplex language — to the slacks. The reduced costs are associated to the decision variables.

8.1.6 Definition. The *Reduced Cost* u_i for an NBV decision variable x_i is the amount the objective value would decrease if x_i would be forced to be 1, and thus a BV — given that the change from $x_i = 0$ to $x_i = 1$ is small.

8.1.7 Remark. Here are some interpretations and remarks of reduced costs that should help you to understand the concept:

- The clause "given that the change from $x_i = 0$ to $x_i = 1$ is small" is a similar clause that the clause "given that the optimal BVs don't change" was in Definition 8.1.1 of shadow price. Indeed, it may be, e.g., that forcing $x_i \geq 1$ will make the LP infeasible. Remember: In sensitivity analysis we are talking about *small changes* — whatever that means. The analysis may, and most often will, fail for big changes.
- Decision variables that are BVs do not have reduced costs, or, if you like, their reduced costs are zero.
- The reduced cost is also known as *Opportunity Cost*. Indeed, suppose we are given the *forced* opportunity (there are no problems — only opportunities) to produce one unit of x_i that we would not otherwise manufacture at all. This opportunity would cost us, since our optimized objective would decrease to a suboptimal value. Indeed, we have now one more constraint — the forced opportunity — in our optimization problem. So, the optimal solution can only get worse. The decrease of the objective value is the opportunity cost.
- The reduced cost u_i of x_i is the amount by which the objective coefficient c_i for x_i needs to change before x_i will become non-zero.
- As an example of the point above, consider that you are producing $x_1, \ldots, x_n$ that will give you profits $c_1, \ldots, c_n$. You have some constraints, but the actual form of them does not matter here. Now, you form the LP to optimize your profit, and you solve it. You get optimal solution for productions: $x_1^*, x_2^*, \ldots, x_n^*$, and you get your optimal profit z^*. You notice that, say, $x_2^* = 0$. So, obviously the profit c_2 for x_2 is not big enough. Then you ask yourself: How big should the profit c_2 for x_2 be so that it becomes more profitable to produce x_2, at least a little, rather than not to produce x_2 at all? The answer is $c_2 + u_2$. This means that the profit must increase at least by the reduced cost before it becomes more profitable to produce a product you would not produce otherwise.

Let us now consider the reduced cost with GLPK with the example:

8.1.8 Example.

$$
\begin{array}{lrcrcrl}
\max z & = & 4x_1 & + & 3x_2 & & (0)\\
\text{s.t.} & & 2x_1 & + & 3x_2 & \leq 16 & (1)\\
& & -3x_1 & + & 2x_2 & \leq 13 & (2)\\
& & & & 2x_2 & \leq 15 & (3)\\
& & 2x_1 & + & x_2 & \leq 4 & (4)\\
& & & & x_1, x_2 & \geq 0 &
\end{array}
$$

The GNU MathProg code for Example 8.1.8 is

```
# Sensitivity analysis for Example 8.1.10
# Part 1 - The original problem

# Decision variables
var x1 >=0;
var x2 >=0;

# Objective
maximize z: 4*x1 + 3*x2;

# Constraints
s.t. r1:  2*x1 + 3*x2 <= 16;
s.t. r2: -3*x1 + 2*x2 <= 13;
s.t. r3:         2*x2 <= 15;
s.t. r4:  2*x1 +   x2 <=  4;

end;
```

The relevant part of the `glpsol` report says

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B             12
     2 r1           B             12                         16
     3 r2           B              8                         13
     4 r3           B              8                         15
     5 r4           NU             4                          4             3

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           NL             0             0                          -2
     2 x2           B              4             0
```

So, where are the reduced costs for `x1` and `x2`? They are in the `Marginal` column. So, `glpsol` calls reduced costs and shadow prices with the same name "marginal". How strange! Well, it is actually not at all so strange once we have learnt about duality. For now, let us just accept this as a mystery to be solved later. Back to the `glpsol` report: For `x1` there is the reduced cost of 2 (`glpsol` uses a non-standard sign). For the decision variable `x2` there is no reduced cost, since `x2` is a BV.

Let us then test the interpretation

> "reduced cost is the decrease in the value of the objective if we are forced to produce one unit where we otherwise would produce none".

We test the interpretation with the following LP:

$$\begin{array}{lrcrcrl}
\max z & = & 4x_1 & + & 3x_2 & & (0)\\
\text{s.t.} & & 2x_1 & + & 3x_2 & \leq\ 16 & (1)\\
& & -3x_1 & + & 2x_2 & \leq\ 13 & (2)\\
& & & & 2x_2 & \leq\ 15 & (3)\\
& & 2x_1 & + & x_2 & \leq\ 4 & (4)\\
& & & & x_1 & \geq\ 1 & (5)\\
& & & & x_1, x_2 & \geq\ 0 &
\end{array}$$

So, we have added to the LP of Example 8.1.8 the requirement that we must have at least one x_1 in the solution. This is the constraint (5). Remember that without this requirement we would have zero x_1's in the solution.

So, here is the GNU MathProg code for this problem:

```
# Sensitivity analysis for Example 8.1.10
# Part 2 - x1 forced to be at least one

# Decision variables
var x1 >=0;
var x2 >=0;

# Objective
maximize z: 4*x1 + 3*x2;

# Constraints
s.t. r1:  2*x1 + 3*x2 <= 16;
s.t. r2: -3*x1 + 2*x2 <= 13;
s.t. r3:         2*x2 <= 15;
s.t. r4:  2*x1 +   x2 <=  4;
s.t. r5:           x1 >=  1;

end;
```

Here is the `glpsol` report:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B             10
     2 r1           B              8                          16
     3 r2           B              1                          13
     4 r3           B              4                          15
     5 r4           NU             4                           4             3
     6 r5           NL             1             1                          -2

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x1           B              1             0
     2 x2           B              2             0
```

We see that the interpretation is indeed correct: The previous optimal value 12 dropped by 2 into 10.

Sensitivity Analysis with Simplex with Matrices

We have learned how to perform sensitivity analysis with GLPK. Let us then consider how sensitivity analysis works with the Simplex tableaux. To be more precise we explain how the shadow prices and the reduced costs can be read from the final optimal Simplex tableau. It would actually be very easy to just to give you the answer. Instead, we will try to explain why the answer is what it is. So we shall revisit the Simplex algorithm, now in matrix form. The answer to the critical question

> "how to read shadow prices and reduced costs from the optimal Simplex tableau"

is deferred to the end of the subsection.

Before we start let us introduce some matrix notation for the Simplex method. In doing this we also explain a little why the Simplex method works. It is best, at least here in the beginning, to consider an example:

8.1.9 Example. Consider the Giapetto's LP 3.3.1 in slack form:

$$\begin{array}{lrcrcrcrcrcr} \max & z & = & 3x_1 & + & 2x_2 & & & & & & \\ \text{s.t.} & & & 2x_1 & + & x_2 & + & s_1 & & & = & 100 \\ & & & x_1 & + & x_2 & & & + & s_2 & = & 80 \\ & & & x_1 & & & & & & + \; s_3 & = & 40 \\ & & & & & & & x_1, x_2, s_1, s_2, s_3 & & & \geq & 0 \end{array}$$

Define

$$\mathbf{c} = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 100 \\ 80 \\ 40 \end{bmatrix}.$$

Then the Giapetto's slack form LP in Example 8.1.9 can be written neatly in the matrix form as

$$\begin{array}{lrcl} \max & z & = & \mathbf{c}'\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \tag{8.1.10}$$

Now, the first Simplex tableau for Example 8.1.9 is

Row	z	x_1	x_2	s_1	s_2	s_3	BV	RHS
0	1	−3	−2	0	0	0	$z =$	0
1	0	2	1	1	0	0	$s_1 =$	100
2	0	1	1	0	1	0	$s_2 =$	80
3	0	1	0	0	0	1	$s_3 =$	40

This tableau corresponds to the augmented matrix

$$\left[\begin{array}{cc|c} 1 & -\mathbf{c}' & 0 \\ \mathbf{0} & \mathbf{A} & \mathbf{b} \end{array}\right].$$

The last (optimal) Simplex tableau is

Row	z	x_1	x_2	s_1	s_2	s_3	BV	RHS
0	1	0	0	1	1	0	$z =$	180
1	0	0	1	−1	2	0	$x_2 =$	60
2	0	0	0	−1	1	1	$s_3 =$	20
3	0	1	0	0	−1	0	$x_1 =$	20

This tableau corresponds to the augmented matrix

$$\left[\begin{array}{ccc|c} 1 & -\mathbf{c}^*_{\mathrm{BV}}{}' & -\mathbf{c}^*_{\mathrm{NBV}}{}' & z^* \\ \mathbf{0} & \mathbf{B}^* & \mathbf{N}^* & \mathbf{x}^*_{\mathrm{BV}} \end{array}\right], \tag{8.1.11}$$

where we have used the denotations

$$\begin{aligned}
\mathbf{x}^*_{\mathrm{BV}} &= \begin{bmatrix} x_2 \\ s_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 60 \\ 20 \\ 20 \end{bmatrix} \\
&= \text{the BV component of the optimal BFS } \mathbf{x}^*, \\
\mathbf{x}^*_{\mathrm{NBV}} &= \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
&= \text{the NBV component of the optimal BFS } \mathbf{x}^*, \\
\mathbf{B}^* &= \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \text{the BV columns of} \\
&\quad \text{the “augmented constraint matrix at optimum”}, \\
\mathbf{N}^* &= \begin{bmatrix} -1 & 2 \\ -1 & 1 \\ 0 & -1 \end{bmatrix} \\
&= \text{the NBV columns of} \\
&\quad \text{the “augmented constraint matrix at optimum”}, \\
\mathbf{c}^*_{\mathrm{BV}} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\
&= \text{the BV coefficients at optimum}, \\
\mathbf{c}^*_{\mathrm{NBV}} &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \text{the NBV coefficients at optimum.}
\end{aligned}$$

8.1.12 Remark. The columns in (8.1.11) are in different order than in the corresponding optimal Simplex tableau. This is unfortunate, but cannot be avoided, since we want to list first the BVs and then the NBVs. This is not a problem, however. Indeed, it is obvious that the original optimal tableau

Row	z	x_1	x_2	s_1	s_2	s_3	BV	RHS
0	1	0	0	1	1	0	$z =$	180
1	0	0	1	−1	2	0	$x_2 =$	60
2	0	0	0	−1	1	1	$s_3 =$	20
3	0	1	0	0	−1	0	$x_1 =$	20

and the column-changed tableau

Row	z	x_2	s_3	x_1	s_1	s_2	BV	RHS
0	1	0	0	0	1	1	$z =$	180
1	0	1	0	0	−1	2	$x_2 =$	60
2	0	0	1	0	−1	1	$s_3 =$	20
3	0	0	0	1	0	−1	$x_1 =$	20

are the same. And this latter tableau should look very similar to the augmented matrix (8.1.11), indeed.

Now, using the obvious non-starred analog of the notation we have just introduced the matrix form LP (8.1.10) can be stated in matrix form as

$$\begin{array}{lrcl} \max & z = \mathbf{c}'_{\mathrm{BV}}\mathbf{x}_{\mathrm{BV}} + \mathbf{c}'_{\mathrm{NBV}}\mathbf{x}_{\mathrm{NBV}} & & \\ \text{s.t.} & \mathbf{B}\mathbf{x}_{\mathrm{BV}} + \mathbf{N}\mathbf{x}_{\mathrm{NBV}} & = & \mathbf{b} \\ & \mathbf{x}_{\mathrm{BV}}, \mathbf{x}_{\mathrm{NBV}} & \geq & \mathbf{0} \end{array} \tag{8.1.13}$$

In augmented matrix form (8.1.13) reads

$$\left[\begin{array}{ccc|c} 1 & -\mathbf{c}'_{\mathrm{BV}} & -\mathbf{c}'_{\mathrm{NBV}} & 0 \\ \mathbf{0} & \mathbf{B} & \mathbf{N} & \mathbf{b} \end{array}\right]. \tag{8.1.14}$$

(Note that we are now back in the initial problem, or the first tableau.)

Now, we show how the last simplex tableau is found analytically from the LP (8.1.13), or from the first simplex tableau, which is nothing but the augmented matrix form (8.1.14). First, we multiply the constraint equation from the left by the inverse $\mathbf{B}^{-1}$. At this point you may wonder how does one find the inverse $\mathbf{B}^{-1}$. Well, actually we found it in the Simplex Step 5, where we solved the tableau. So, the Step 5 was actually finding the inverse of the BV part $\mathbf{B}$ of the matrix $\mathbf{A}$. So, the constraint

$$\mathbf{B}\mathbf{x}_{\mathrm{BV}} + \mathbf{N}\mathbf{x}_{\mathrm{NBV}} = \mathbf{b}$$

becomes

$$\mathbf{x}_{\mathrm{BV}} + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathrm{NBV}} = \mathbf{B}^{-1}\mathbf{b}.$$

Now, recall that the NBVs are all zeros. So, $\mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathrm{NBV}} = \mathbf{0}$, and we actually have the constraint

$$\mathbf{x}_{\mathrm{BV}} = \mathbf{B}^{-1}\mathbf{b}.$$

What have we found out? We have found out that the optimization problem (8.1.13), that was equivalent to the problem (8.1.10), is actually equivalent to the problem

$$\begin{array}{lrcl} \max & z = \mathbf{c}'_{\mathrm{BV}}\mathbf{x}_{\mathrm{BV}} + \mathbf{c}'_{\mathrm{NBV}}\mathbf{x}_{\mathrm{NBV}} & & \\ \text{s.t.} & \mathbf{x}_{\mathrm{BV}} & = & \mathbf{B}^{-1}\mathbf{b} \\ & \mathbf{x}_{\mathrm{BV}}, \mathbf{x}_{\mathrm{NBV}} & \geq & \mathbf{0} \end{array} \tag{8.1.15}$$

Now, look at the objective row "constraint"

$$z = \mathbf{c}'_{\mathrm{BV}}\mathbf{x}_{\mathrm{BV}} + \mathbf{c}'_{\mathrm{NBV}}\mathbf{x}_{\mathrm{NBV}}.$$

With a little bit of matrix algebra we see that this is equivalent to

$$z = \mathbf{c}'_{\mathrm{BV}}\mathbf{B}^{-1}\mathbf{b}.$$

Indeed, the $\mathbf{c}'_{\text{NBV}}\mathbf{x}_{\text{NBV}}$ part vanishes, since $\mathbf{x}_{\text{NBV}} = \mathbf{0}$, and we know already that $\mathbf{x}_{\text{BV}} = \mathbf{B}^{-1}\mathbf{b}$. So, we have found that the LP (8.1.10), or the LP (8.1.13), or the LP (8.1.15), is actually equivalent to the LP

$$\begin{array}{lrcl} \max & z = \mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}\mathbf{b} & & \\ \text{s.t.} & \mathbf{x}_{\text{BV}} & = & \mathbf{B}^{-1}\mathbf{b} \\ & \mathbf{x}_{\text{BV}}, \mathbf{x}_{\text{NBV}} & \geq & \mathbf{0} \end{array} \tag{8.1.16}$$

It may not be obvious at this point, but we have actually seen the vector $\mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}$ — or actually the vector $-\mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}$ — already. They are the shadow prices:

$$\pi' \quad = \quad \mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}.$$

Actually, defined this way, the shadow prices are associated to all BVs, not only to BVs that are slacks, and thus related to the constraints only.

Let us now collect what we have found out — at least to some extend — of the Simplex algorithm as a theorem.

8.1.17 Theorem. *Let* $\mathbf{x}_{\text{BV}}$ *be a BFS of the LP (8.1.10). Then*

$$\mathbf{x}_{\text{BV}} \quad = \quad \mathbf{B}^{-1}\mathbf{b},$$

and the corresponding value of the objective is

$$z \quad = \quad \mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}\mathbf{b} \quad = \quad \pi'\mathbf{b}.$$

Moreover, the coefficients in the 0*th row of the NBV part of the Simplex tableau are* $\pi'\mathbf{N} - \mathbf{c}'_{\text{NBV}}$.

The point $\mathbf{x}$ *is feasible if* $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, *and the point* $\mathbf{x}$ *is optimal if it is feasible and the reduced cost vector satisfies* $\pi'\mathbf{N} - \mathbf{c}'_{\text{NBV}} \geq \mathbf{0}'$.

Now, finally, comes the answer you have been waiting for. Well, not yet! We still need some notation. We shall split the coefficients in the final Simplex tableau in two parts. Here we also use different notation for decision variables and slacks: previously the vector $\mathbf{x}$ contained both the decision variables and the slack. Now, the vector $\mathbf{x}$ will denote only the decision variables, and the slacks (and excesses) will be denoted by the vector $\mathbf{s}$. The matrix $\mathbf{A}$ will still denote the extended matrix that includes the slacks.

Let

$$\begin{array}{rcl} \mathbf{u} & = & \text{coefficients of the decision variables at the 0th row,} \\ \mathbf{v} & = & \text{coefficients of the slacks at the 0th row.} \end{array}$$

Now, we have just learned that

$$\begin{array}{l} \text{if the decision variable } x_j \text{ is BV, then } u_j = 0, \\ \text{if the decision variable } x_j \text{ is NBV, then } u_j = \pi'\mathbf{a}_{\bullet j} - c_j. \end{array}$$

We have also just learned that

$$\begin{aligned}&\text{if the slack } s_i \text{ is BV, then } v_i = 0,\\ &\text{if the slack } s_i \text{ is NBV, then } v_i = \pi' \mathbf{a}_{\bullet(n+i)} \quad = \quad \pi_i.\end{aligned}$$

With the notation introduced above, the 0th row of the optimal Simplex tableau represents the equation

$$z + \sum_{j=1}^{n} u_j x_j + \sum_{i=1}^{m} v_i s_i \quad = \quad z^*, \tag{8.1.18}$$

or — if you like matrix notation — the equation

$$z + \mathbf{u}'\mathbf{x} + \mathbf{v}'\mathbf{s} \quad = \quad z^*.$$

Suppose then that we change the situation so that a NBV x_k becomes a BV with value $x_k = 1$. This of course requires changing the values of BVs, but the bottom line is that the equation (8.1.18) will now have an additional term $u_k x_k = u_k$ on its LHS (Left Hand Side). This is the only change in the 0th row since the BVs had coefficients 0 in the optimal 0th row, as they were eliminated from the 0th row by the Simplex algorithm. Consequently the RHS has decreased by u_k. The conclusion is that

u_k is the reduced cost of the NBV x_k.

Suppose then that some resource constraint b_k is increased to $b_k + 1$. Since $\mathbf{c}_{\text{BV}}$ and $\mathbf{B}^{-1}$ remain unchanged, remains $\pi = \mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}$ also unchanged. So, the new optimum must be

$$\begin{aligned} z^{\text{new}*} &= \pi' \mathbf{b}^{\text{new}} \\ &= \sum_{i=1}^{n} \pi_i b_i^{\text{new}} \\ &= \sum_{i=1, i\neq k}^{n} \pi_i b_i \;+\; \pi_k (b_k + 1) \\ &= \sum_{i=1}^{n} \pi_i b_i + \pi_k \\ &= z^{\text{old}*} + \pi_k \\ &= z^{\text{old}*} + v_k. \end{aligned}$$

The conclusion is that

v_k is the shadow price of the resource b_k.

(Until now the association $\pi' = \mathbf{c}'_{\text{BV}}\mathbf{B}^{-1}$ was just notation. Now we have just justified that if π is defined this way, then it actually is the vector of shadow prices in the sense of Definition 8.1.1.)

So, back to the Giapetto's problem 8.1.9 with the final optimal Simplex tableau:

Row	z	x_1	x_2	s_1	s_2	s_3	BV	RHS
0	1	0	0	1	1	0	$z =$	180
1	0	0	1	−1	2	0	$x_2 =$	60
2	0	0	0	−1	1	1	$s_3 =$	20
3	0	1	0	0	−1	0	$x_1 =$	20

For the 0th row we read that

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{v} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

So, the shadow prices for the NBV slacks s_1 and s_2 are 1 for both. So, additional carpentry and finishing hours are worth 1 Euro per hour both for Giapetto. Since s_3 is a non-zero BV additional unit of market demand for soldiers is worthless to Giapetto. Indeed, in the optimal solution Giapetto is not meeting the market demand. Finally, we see that the reduced costs are zero, since all the decision variables are BVs.

Let us end this section with yet another example that illustrates how shadow prices and reduced costs can be read from the optimal Simplex tableau. The following Example is actually the Dakota Furniture's problem 6.2.2 without the market demand constraint $x_2 \leq 5$ (that turned out to be redundant anyway).

8.1.19 Example. We want to perform sensitivity analysis to the LP

$$\begin{array}{lrcrcrcr} \max & z & = & 60x_1 & + & 30x_2 & + & 20x_3 & \\ \text{s.t.} & & & 8x_1 & + & 6x_2 & + & x_3 & \leq & 48 \\ & & & 4x_1 & + & 2x_2 & + & 1.5x_3 & \leq & 20 \\ & & & 2x_1 & + & 3x_2 & + & 0.5x_3 & \leq & 8 \\ & & & & & x_1, x_2, s_1, s_2, s_3 & & & \geq & 0 \end{array}$$

by using the Simplex method.

We see that the LP in Example 8.1.19 Has only inequalities of type $\leq$, and all the RHSs are non-negative. So, we will only have surplus type slacks, and

we obtain immediately a canonical slack form. So, our first simplex tableau is (was)

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	BV	RHS
0	1	-60	-30	-20	0	0	0	$z =$	0
1	0	8	6	1	1	0	0	$s_1 =$	48
2	0	4	2	1.5	0	1	0	$s_2 =$	20
3	0	2	3	0.5	0	0	1	$s_3 =$	8

Next we have to perform the Simplex algorithm to find the optimal tableau. After long and tedious tableau-dancing in Chapter 6 we obtained the optimal tableau:

Row	z	x_1	x_2	x_3	s_1	s_2	s_3	BV	RHS
0	1	0	5	0	0	10	10	$z =$	280
1	0	0	-2	0	1	2	-8	$s_1 =$	24
2	0	0	-2	1	0	2	-4	$x_3 =$	8
3	0	1	1.25	0	0	-0.5	1.5	$x_1 =$	2

Now we can read sensitivity information from the 0th row:

$$\mathbf{u} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{v} = \begin{bmatrix} 0 \\ 10 \\ 10 \end{bmatrix}.$$

We see that the reduced cost for the not-produced product x_2 (tables) is 5. This means, e.g., that the profit for making tables should increase at least €5 before it makes sense to produce them. Or, if you like, producing one table would decrease the profit €280 by €5. The reduced costs for x_1 and x_3 Âăare zero, since they are BVs. The shadow prices are: 0 for the slack s_1 (lumber), since it is not active, and 10 for both the active constraints s_2 and s_3 (finishing and carpentry). So, additional carpentry and finishing hours are both worth €10 for Dakota and additional lumber is worthless.

8.2 Dual Problem

Finding Dual

Associated with any LP there is another LP, called the *dual* — and then the original LP is called the *primal*. The relationship between the primal and dual is important because it gives interesting economic insights. Also, it is important because it gives a connection between the shadow prices and the reduced costs.

In general, if the primal LP is a maximization problem, the dual is a minimization problem — and vice versa. Also, the constraints of the primal LP are the coefficients of the objective of the dual problem — and vice versa. If

the constraints of the primal LP are of type $\leq$ then the constraints of the dual LP are of type $\geq$ — and vice versa.

Let us now give the formal definition of the dual. We assume that the primal LP is in standard form of Definition 5.1.9. Since all LPs can be transformed into a standard form this assumption does not restrict the generality of the duality. The assumption is made only for the sake of convenience.

8.2.1 Definition. The *dual* of the standard form LP

$$
\begin{array}{lcccccccccccc}
\max z & = & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & & \\
\text{s.t.} & & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \leq & b_1, \\
 & & a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & \leq & b_2, \\
 & & \vdots & & \vdots & & \ddots & & \vdots & \vdots & \vdots \\
 & & a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \leq & b_m, \\
 & & & & & & & & x_1, x_2, \ldots, x_n & \geq & 0.
\end{array}
\tag{8.2.2}
$$

is

$$
\begin{array}{lcccccccccccc}
\min w & = & b_1y_1 & + & b_2y_2 & + & \cdots & + & b_my_m & & \\
\text{s.t.} & & a_{11}y_1 & + & a_{21}y_2 & + & \cdots & + & a_{m1}y_m & \geq & c_1, \\
 & & a_{11}y_1 & + & a_{22}y_2 & + & \cdots & + & a_{m2}x_m & \geq & c_2, \\
 & & \vdots & & \vdots & & \ddots & & \vdots & \vdots & \vdots \\
 & & a_{1n}y_1 & + & a_{2n}y_2 & + & \cdots & + & a_{mn}y_m & \geq & c_n, \\
 & & & & & & & & y_1, y_2, \ldots, y_m & \geq & 0.
\end{array}
\tag{8.2.3}
$$

In matrix form the duality can be written as: The dual of the LP

$$
\begin{array}{lrcl}
\max & z & = & \mathbf{c}'\mathbf{x} \\
\text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\
 & \mathbf{x} & \geq & \mathbf{0}
\end{array}
$$

is

$$
\begin{array}{lrcl}
\min & w & = & \mathbf{b}'\mathbf{y} \\
\text{s.t.} & \mathbf{A}'\mathbf{y} & \geq & \mathbf{c} \\
 & \mathbf{y} & \geq & \mathbf{0}
\end{array}
$$

8.2.4 Example. Consider the LP

$$\max z = [1\ 2\ 3]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\text{s.t.} \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 10 \\ 11 \end{bmatrix}.$$

The dual LP is

$$\min w = [10\ 11]\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\text{s.t.} \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

8.2.5 Remark. Let us discuss briefly about concept of duality in general and the duality of Definition 8.2.1 in particular.

- In general, dual is a transformation with the following property: *Transforming twice you get back.* This is the abstract definition of duality. In mathematics a function is f is called *involution* if it is its own inverse, i.e., $f^{-1} = f$. So, duality is a *meta-mathematical involution.*
- Looking at Definition 8.2.1 one sees the dual is LP itself. So, it can be transformed into a standard form, and the one can construct the dual of the dual. When one does so one gets back to the original primal LP, i.e., *the dual of the dual is primal.* So, the dual of Definition 8.2.1 deserves its name.
- We have already seen one duality between LPs before: A minimization problem is in duality with a maximization problem with the transform where the objective function is multiplied by -1. The usefulness of this simple duality was that we only need to consider maximization problems, and the solution of the minimization problem is -1 times the solution of the corresponding maximization problem in this simple duality. Also, the optimal decisions in the maximization and minimization problems are the same.
- The duality of Definition 8.2.1 is more complicated than the simple "multiply by -1 duality" of the previous point. This makes the duality of Definition 8.2.1 in some sense more useful than the simple "multiply by -1 duality". Indeed, since the transformation is more complicated, our

change of perspective is more radical, and thus this transformation gives us better intuition of the original problem.

- The duality of Definition 8.2.1 is very useful because of the following theorems: The *weak duality theorem* states that the objective function value w of the dual at any feasible solution $\mathbf{y}$ is always greater than or equal to the objective function value z of the primal at any feasible solution $\mathbf{x}$:

$$w = \mathbf{b}'\mathbf{y} \quad \geq \quad \mathbf{c}'\mathbf{x} = z.$$

The weak duality theorem can be used to get upper bounds to the primal LP. The *strong duality theorem* states that if the primal has an optimal solution, $\mathbf{x}^*$, then the dual also has an optimal solution, $\mathbf{y}^*$, such that

$$z^* = \mathbf{c}'\mathbf{x}^* \quad = \quad \mathbf{b}'\mathbf{y}^* = w^*.$$

The strong duality theorem can be used to solve the primal LP. We shall prove the weak and strong duality theorems later in this lecture.

Let us find a dual of an LP that is not in standard form.

8.2.6 Example. Consider the LP

$$\begin{array}{lrcrcrcrcr}
\min & z & = & 50x_1 & + & 20x_2 & + & 30x_3 & & \\
\text{s.t.} & & & 2x_1 & + & 3x_2 & + & 4x_3 & \geq & 11 \\
& & & 12x_1 & + & 13x_2 & + & 14x_3 & \leq & 111 \\
& & & x_1 & + & x_2 & + & x_3 & = & 1 \\
& & & & & & & x_1, x_2, x_3 & \geq & 0
\end{array}$$

The LP of Example 8.2.6 is not in standard form. So, before constructing its dual, we transform it into standard form. This is not necessary. Sometimes we can be clever, and find the dual without first transforming the primal into standard form. But we don't feel clever now. So, here is the standard form:

$$\begin{array}{lrcrcrcrcr}
\max & -z & = & -50x_1 & - & 20x_2 & - & 30x_3 & & \\
\text{s.t.} & & & -2x_1 & - & 3x_2 & - & 4x_3 & \leq & -11 \\
& & & 12x_1 & + & 13x_2 & + & 14x_3 & \leq & 111 \\
& & & x_1 & + & x_2 & + & x_3 & \leq & 1 \\
& & & -x_1 & - & x_2 & - & x_3 & \leq & -1 \\
& & & & & & & x_1, x_2, x_3 & \geq & 0
\end{array}$$

Now we are ready to present the dual:

$$\begin{array}{lrcrcrcrcrcr}
\min & -w & = & -11y_1 & + & 111y_2 & + & y_3 & - & y_4 & & \\
\text{s.t.} & & & -2y_1 & + & 12y_2 & + & y_3 & - & y_4 & \geq & -50 \\
& & & -3y_1 & + & 13y_2 & + & y_3 & - & y_4 & \geq & -20 \\
& & & -4y_1 & + & 14y_2 & + & y_3 & - & y_4 & \geq & -30 \\
& & & & & & & y_1, y_2, y_3, y_4 & & & \geq & 0
\end{array} \tag{8.2.7}$$

(we used variable $-w$ in the dual because there was variable $-z$ in the standard form primal). Note now that th dual LP (8.2.7) in in "dual standard form": It is a minimization problem with only inequalities of type $\geq$. The original primal LP was a minimization problem. So, it is natural to express the dual LP as a maximization problem. Also, inequalities of type $\leq$ are more natural to maximization problems than the opposite type inequalities $\geq$. So, let us transform the dual LP (8.2.7) into a maximization problem with $\leq$ type inequalities. In fact, let us transform the dual LP (8.2.7) into a standard form. We obtain

$$\begin{array}{lrcrcrcrcr} \max & w & = & 11y_1 & - & 111y_2 & - & y_3 & + & y_4 & \\ \text{s.t.} & & & 2y_1 & - & 12y_2 & - & y_3 & + & y_4 & \leq & 50 \\ & & & 3y_1 & - & 13y_2 & - & y_3 & + & y_4 & \leq & 20 \\ & & & 4y_1 & - & 14y_2 & - & y_3 & + & y_4 & \leq & 30 \\ & & & & & & & y_1, y_2, y_3, y_4 & & & \geq & 0 \end{array}$$

Economic Interpretation of Dual

Let us recall — again — the Dakota Furniture's problem 6.2.2 (without the market demand constraint that turned out to be irrelevant anyway):

$$\begin{array}{lrcrcrcrcrl} \max & z & = & 60x_1 & + & 30x_2 & + & 20x_3 & & & \\ \text{s.t.} & & & 8x_1 & + & 6x_2 & + & x_3 & \leq & 48 & \text{(lumber)} \\ & & & 4x_1 & + & 2x_2 & + & 1.5x_3 & \leq & 20 & \text{(finishing)} \\ & & & 2x_1 & + & 1.5x_2 & + & 0.5x_3 & \leq & 8 & \text{(carpentry)} \\ & & & & & & & x_1, x_2, x_3 & \geq & 0 & \end{array}$$

where

$$\begin{array}{rcl} x_1 & = & \text{number of desks manufactured} \\ x_2 & = & \text{number of tables manufactured} \\ x_3 & = & \text{number of chairs manufactured} \end{array}$$

Now, the dual of this problem is

$$\begin{array}{lrcrcrcrcrl} \min & w & = & 48y_1 & + & 20y_2 & + & 8y_3 & & & \\ \text{s.t.} & & & 8y_1 & + & 4y_2 & + & 2y_3 & \geq & 60 & \text{(desk)} \\ & & & 6y_1 & + & 2y_2 & + & 1.5y_3 & \geq & 30 & \text{(table)} \\ & & & x_1 & + & 1.5y_2 & + & 0.5y_3 & \geq & 20 & \text{(chair)} \\ & & & & & & & y_1, y_2, y_3 & \geq & 0 & \end{array} \tag{8.2.8}$$

We have given the constraints the names (desk), (table), and (chair). Those were the decision variables x_1, x_2 and x_3 in the primal LP. By symmetry, or duality, we could say that y_1 is associated with lumber, y_2 with finishing, and y_3 with carpentry. What is going on here? It is instructive to represent the

data of the Dakota's problem in a table where we try to avoid taking Dakota's point of view:

	Desk	**Table**	**Chair**	***Availability***
Lumber	8 units	6 units	1 unit	48 units
Finishing	4 hours	2 hours	1.5 hours	20 hours
Carpentry	2 hours	1.5 hours	0.5 hours	8 hours
Price	€60	€30	€20	

Now, the table above can be read either horizontally of vertically. You should already know how the read the table above horizontally. That is the Dakota's point of view. But what does it mean to read the table vertically? Here is the explanation, that is also the economic interpretation of the dual LP:

Suppose you are an entrepreneur who wants to purchase all of Dakota's resources — maybe you are a competing furniture manufacturer, or maybe you need the resources to produce soldiers and trains like Giapetto. Then you must determine the price you are willing to pay for a unit of each of Dakota's resources. But what are the Dakota's resources? Well they are lumber, finishing hours, and carpentry hours, that Dakota uses to make its products. So, the decision variables for the entrepreneur who wants to buy Dakota's resources are:

$$\begin{aligned} y_1 &= \text{price to pay for one unit of lumber} \\ y_2 &= \text{price to pay for one hour of finishing labor} \\ y_3 &= \text{price to pay for one hour of carpentry labor} \end{aligned}$$

Now we argue that the resource prices y_1, y_2, y_3 should be determined by solving the Dakota dual (8.2.8).

First note that you are buying *all* of Dakota's resources. Also, note that this is a minimization problem: You want to pay as little as possible. So, the objective function is

$$\min w \quad = \quad 48y_1 + 20y_2 + 8y_3.$$

Indeed, Dakota has 48 units of lumber, 20 hours of finishing labor, and 8 hours of carpentry labor.

Now we have the decision variables and the objective. How about constraints? In setting the resource prices y_1, y_2, and y_3, what kind of constraints do you face? You must set the resource prices high enough so that Dakota would sell them to you. Now Dakota can either use the resources itself, or sell them to you. How is Dakota using its resources? Dakota manufactures desks, tables, and chair. Take desks first. With 8 units of lumber, 4 hours of finishing labor, and 2 hours of carpentry labor Dakota can make a desk that will sell for €60. So, you have to offer more than €60 for this particular combination of resources. So, you have the constraint

$$8y_1 + 4y_2 + 2y_3 \quad \geq \quad 60.$$

But this is just the first constraint in the Dakota dual, denoted by (desk). Similar reasoning shows that you must pay at least €30 for the resources Dakota uses to produce one table. So, you get the second constraint, denoted by (table), of the Dakota dual:

$$6y_1 + 2y_2 + 1.5y_3 \quad \geq \quad 30.$$

Similarly, you must offer more than €20 for the resources the Dakota can use itself to produce one chair. That way you get the last constraint, labeled as (chair), of the Dakota dual:

$$y_1 + 1.5y_2 + 0.5y_3 \quad \geq \quad 20.$$

We have just interpreted economically the dual of a maximization problem. Let us then change our point of view to the opposite and interpret economically the dual of a minimization problem.

8.2.9 Example. My diet requires that all the food I eat come from the four "basic food groups": chocolate cake, ice cream, soda, and cheese cake. At present four foods are available: brownies, chocolate ice cream, cola, and pineapple cheesecake. The costs of the foods (in Cents) and my daily nutritional requirements together with their calorie, chocolate, sugar, and fat contents are listed in the table below this box.

I want to minimize the cost of my diet. How should I eat?

	Calories	**Chocolate**	**Sugar**	**Fat**	***Price***
Brownie	400	3	2	2	50
Chocolate ice cream	200	2	2	4	20
Cola	150	0	4	1	30
Pineapple cheesecake	500	0	4	5	80
Requirement	500	6	10	8	

Let us then find the LP for the Diet problem of Example 8.2.9. As always, we first determine the decision variables. The decision to be made is: how much each type of food should be eaten daily. So, we have the decision variables

x_1 = number of brownies eaten daily,
x_2 = number (of scoops) of chocolate ice creams eaten daily,
x_3 = number (of bottles) of cola drunk daily,
x_4 = number (of pieces) of pineapple cheesecake eaten daily.

Next we define the objective. We want to minimize the cost of the diet. So, the objective is

$$\min z \quad = \quad 50x_1 + 20x_2 + 30x_3 + 80x_4.$$

Finally, we define the constraints. The daily calorie intake requirement gives

$$400x_1 + 200x_2 + 150x_3 + 500x_4 \quad \geq \quad 500.$$

The daily chocolate requirement gives

$$3x_1 + 2x_2 \quad \geq \quad 6.$$

The daily sugar requirement gives

$$2x_1 + 2x_2 + 4x_3 + 4x_4 \quad \geq \quad 10,$$

and the daily fat requirement gives

$$2x_1 + 4x_2 + x_3 + 5x_4 \quad \geq \quad 8.$$

So, we see that the Diet problem of Example 8.2.9 is the LP

(8.2.10)

$$\begin{array}{lrcrcrcrcll}
\min z = & 50x_1 & + & 20x_2 & + & 30x_3 & + & 80x_4 & & & \\
\text{s.t.} & 400x_1 & + & 200x_2 & + & 150x_3 & + & 500x_4 & \geq & 400 & \text{(calorie)} \\
& 3x_1 & + & 2x_2 & & & & & \geq & 6 & \text{(chocolate)} \\
& 2x_1 & + & 2x_2 & + & 4x_3 & + & 4x_4 & \geq & 10 & \text{(sugar)} \\
& 2x_1 & + & 4x_2 & + & x_3 & + & 5x_4 & \geq & 8 & \text{(fat)} \\
& & & & & x_1, x_2, x_3, x_4 & & & \geq & 0 &
\end{array}$$

What about the dual of (8.2.10) then. Now, the LP (8.2.10) is not in standard form. So, in principle we should first transform it into standard form, and then construct the dual. We shall not do that, however. Instead, we remember that the dual of the dual is primal. So, we read the Definition 8.2.1 backwards, and obtain immediately the dual of (8.2.10):

(8.2.11)

$$\begin{array}{lrcrcrcrcll}
\max w = & 500y_1 & + & 6y_2 & + & 10y_3 & + & 8y_4 & & & \\
\text{s.t.} & 400y_1 & + & 3y_2 & + & 2y_3 & + & 2y_4 & \leq & 50 & \text{(brownie)} \\
& 200y_1 & + & 2y_2 & + & 2y_3 & + & 4y_4 & \leq & 20 & \text{(ice cream)} \\
& 150y_1 & & & + & 4y_3 & + & y_4 & \leq & 30 & \text{(soda)} \\
& 500y_1 & & & + & 4y_3 & + & 5y_4 & \leq & 80 & \text{(cheesecake)} \\
& & & & & y_1, y_2, y_3, y_4 & & & \geq & 0 &
\end{array}$$

What is then the economic interpretation of this dual? Well, reading the table

	Calories	**Chocolate**	**Sugar**	**Fat**	***Price***
Brownie	400	3	2	2	50
Chocolate ice cream	200	2	2	4	20
Cola	150	0	4	1	30
Pineapple cheesecake	500	0	4	5	80
Requirement	500	6	10	8	

vertically, instead of horizontally, we see that we can consider a "nutrient" salesperson who sells calories, chocolate, sugar, and fat. The salesperson wishes to ensure that a dieter will meet all of his daily requirements by purchasing calories, sugar, fat, and chocolate from the the salesperson. So, the salesperson must determine the prices of her products:

$$\begin{aligned} y_1 &= \text{price of a calorie,} \\ y_2 &= \text{price of a unit of chocolate,} \\ y_3 &= \text{price of a unit of sugar,} \\ y_4 &= \text{price of a unit of fat.} \end{aligned}$$

The salesperson wants to maximize her profit. So, what is the salesperson selling? She is selling daily diets. So, the objective is

$$\max w \quad = \quad 500y_1 + 6y_2 + 10y_3 + 8y_4.$$

What are the constraints for the salesperson? In setting the nutrient prices she must set the prices low enough so that it will be in the dieter's economic interest to purchase as his nutrients from her. For example, by purchasing a brownie for €0.50, the dieter can obtain 400 calories, 3 units of chocolate, 2 units of sugar, and 2 units of fat. So, the salesperson cannot charge more than €0.50 for this combination of nutrients. This gives her the brownie constraint

$$400y_1 + 3y_2 + 2y_3 + 2y_4 \quad \leq \quad 50$$

(remember, we counted in Cents). In the similar way the salesperson will have the ice cream, soda, and cheesecake constraints listed in the dual LP (8.2.11).

Duality Theorem

In this subsection we discuss one of the most important results in linear programming: the Duality Theorem. In essence, the Duality Theorem states that the primal and the dual have equal optimal objective function values — given that the problems have optimal solutions. While this result is interesting in its own right, we will see that in proving it we gain many important insights into linear programming.

As before, we assume — for the sake of convenience — that the primal is in standard form. So, the primal will be a maximization problem, and the dual

will be a minimization problem. For the sake of reference you may think that we are dealing with the Dakota's problem 6.2.2 (without the irrelevant market demand constraint) and its dual (8.2.8).

The next theorem is the *Weak Duality Theorem*.

8.2.12 Theorem. *Let* $\mathbf{x}$ *be any BFS of the primal LP and let* $\mathbf{y}$ *be any BFS of the dual LP. Then*

$$z = \mathbf{c}'\mathbf{x} \quad \leq \quad \mathbf{b}'\mathbf{y} = w.$$

Let us actually *prove* the Weak Duality Theorem 8.2.12:

Proof. Consider any of the dual decision variable y_i, $i = 1, \ldots, m$. Since $y_i \geq 0$ we can multiply the ith primal constraint by y_i without changing the direction of the constraint number i. (Moreover, the system remains equivalent, but that's not important here). We obtain

$$y_i a_{i1} x_1 + \cdots + y_i a_{in} x_n \quad \leq \quad b_i y_i \quad \text{for all } i = 1, \ldots, m. \tag{8.2.13}$$

Adding up all the m inequalities (8.2.13), we find that

$$\sum_{i=1}^{m} \sum_{j=1}^{n} y_i a_{ij} x_j \quad \leq \quad \sum_{i=1}^{m} b_i y_i. \tag{8.2.14}$$

Similarly, if we consider any of the primal decision variables x_j, $j = 1, \ldots, n$, we have that $x_j \geq 0$. So, we can multiply the jth dual constraint by the decision x_j without changing the direction of the constraint. We obtain

$$x_j a_{1j} y_1 + \cdots + x_j a_{mj} y_m \quad \geq \quad c_j x_j. \tag{8.2.15}$$

Adding up all the n inequalities (8.2.15), we find that

$$\sum_{i=1}^{m} \sum_{j=1}^{n} y_i a_{ij} x_j \quad \geq \quad \sum_{j=1}^{n} c_j x_j. \tag{8.2.16}$$

Combining (8.2.14) and (8.2.16), we obtain double-inequality

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

But, that's it! □

Let us then consider the consequences — or corollaries — of the Weak Duality Theorem 8.2.12.

8.2.17 Corollary. *If the primal LP and dual LP both are feasible, then the their optimal solutions are bounded.*

Proof. Let $\mathbf{y}$ be a BFS for the dual problem. Then the Weak Duality Theorem 8.2.12 shows that $\mathbf{b}'\mathbf{y}$ is an upper bound for any objective value $\mathbf{c}'\mathbf{x}$ associated with any BFS $\mathbf{x}$ of the primal LP:

$$\mathbf{c}'\mathbf{x} \quad \leq \quad \mathbf{b}'\mathbf{y}.$$

Since this is true for *any* primal decision $\mathbf{x}$, it is true for an optimal primal decision $\mathbf{x}^*$ also. So,

$$\begin{aligned} z^* &= \mathbf{c}'\mathbf{x}^* \\ &= \max\left\{\mathbf{c}'\mathbf{x}\,;\, \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\} \\ &\leq \max\left\{\mathbf{b}'\mathbf{y}\,;\, \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\} \\ &\leq \mathbf{b}'\mathbf{y} \\ &< \infty \end{aligned}$$

is bounded. Now, change the rôles of the primal and the dual, and you see that the claim of Corollary 8.2.17 is true. □

8.2.18 Corollary. *Suppose $\mathbf{x}^*$ is a BFS for the primal and $\mathbf{y}^*$ is a BFS for the dual. Suppose further that $\mathbf{c}'\mathbf{x}^* = \mathbf{b}'\mathbf{y}^*$. Then both $\mathbf{x}^*$ and $\mathbf{y}^*$ are optimal for their respective problems.*

Proof. If $\mathbf{x}$ is any BFS for the primal, then the Weak Duality Theorem 8.2.12 tells us that

$$\mathbf{c}'\mathbf{x} \quad \leq \quad \mathbf{b}'\mathbf{y}^* \quad = \quad \mathbf{c}'\mathbf{x}^*.$$

But this means that $\mathbf{x}^*$ is primal optimal. Now, change the rôles of the primal and dual, and you see that the claim of the Corollary 8.2.18 is true. □

Here is the *Duality Theorem*, or the *Strong Duality Theorem*. To understand the notation, recall Theorem 8.1.17 earlier in this Chapter.

8.2.19 Theorem. *Let*

$$\mathbf{x}_{\mathrm{BV}} = \mathbf{B}^{-1}\mathbf{b}$$

be the optimal BFS to the primal with the corresponding optimal objective value

$$z^* = \mathbf{c}'_{\mathrm{BV}}\mathbf{B}^{-1}\mathbf{b} = \pi'\mathbf{b}.$$

Then π is the optimal BFS for the dual. Also, the values of the objectives at the optimum are the same:

$$w^* \quad = \quad \pi'\mathbf{b} \quad = \quad \mathbf{c}'_{\mathrm{BV}}\mathbf{x}_{\mathrm{BV}}.$$

We shall not prove Theorem 8.2.19 in these notes. Instead, we leave it as an exercise. The author is well aware that this is a very demanding exercise, but not all of the exercises have to be easy! Also, there will be no proofs in the final exam, so it is justified that there is at least one proof in the exercises.

Complementary Slackness

It is possible to obtain an optimal solution to the dual when only an optimal solution to the primal is known using the *Theorem of Complementary Slackness*. To state this theorem, we assume that the primal is in standard form with non-negative RHSs and objective coefficients. The primal decision variables are $\mathbf{x} = [x_1 \ \cdots \ x_n]'$ and the primal slacks are $\mathbf{s} = [s_1 \ \cdots \ s_m]'$. The dual is then a minimization problem with decision variables $\mathbf{y} = [y_1 \ \cdots \ y_m]'$, and with n constraints of type $\geq$ with non-negative RHSs. Let $\mathbf{e} = [e_1 \ \cdots \ e_n]'$ be the excesses of the dual problem associated with the constraints.

So, in slack form the primal LP is

$$\begin{array}{lrcrcrcrl}
\max z & = & c_1x_1 & + \cdots + & c_nx_n & & & & \\
\text{s.t.} & & a_{11}x_1 & + \cdots + & a_{1n}x_n & +s_1 & & & = b_1 \\
& & a_{21}x_1 & + \cdots + & a_{2n}x_n & & +s_2 & & = b_2 \\
& & \vdots & & \vdots & & & \ddots & \vdots \\
& & a_{m1}x_1 & + \cdots + & a_{mn}x_n & & & +s_m & = b_m \\
& & & & & x_1, \ldots, x_n, & s_1, \ldots, s_m & & \geq 0
\end{array}$$

Similarly, the dual LP in slack — or rather excess — form is

$$\begin{array}{lrcrcrcrl}
\min w & = & b_1y_1 & + \cdots + & b_my_m & & & & \\
\text{s.t.} & & a_{11}y_1 & + \cdots + & a_{m1}y_m & -e_1 & & & = c_1 \\
& & a_{12}y_1 & + \cdots + & a_{m2}y_m & & -e_2 & & = c_2 \\
& & \vdots & & \vdots & & & \ddots & \vdots \\
& & a_{1n}y_1 & + \cdots + & a_{mn}y_m & & & -e_n & = c_n \\
& & & & & y_1, \ldots, y_m, & e_1, \ldots, e_n & & \geq 0
\end{array}$$

Here is the *Theorem of Complementary Slackness*.

8.2.20 Theorem. *Let* $\mathbf{x}$ *be a primal BFS, and let* $\mathbf{y}$ *be a dual BFS. Then* $\mathbf{x}$ *is primal optimal and* $\mathbf{y}$ *is dual optimal if and only if*

$$\begin{aligned}
s_iy_i &= 0 \quad \textit{for all} \ \ i = 1, \ldots, m, \\
e_jx_j &= 0 \quad \textit{for all} \ \ j = 1, \ldots, n.
\end{aligned}$$

Before going into the proof of the Complementary Slackness Theorem 8.2.20 let us note that:

8.2.21 Remark. Theorem 8.2.20 says that

> *if a constraint in either the primal or the dual is non-active, then the corresponding variable in the other — complementary — problem must be zero.*

Hence the name *complementary* slackness.

Proof. The theorem 8.2.20 is of the type "if and only if". So, there are two parts in the proof: the "if part" and the "only if part". Before going to those parts let us note that

$$s_i = 0 \quad \text{if and only if} \quad \sum_{j=1}^{m} a_{ij}x_j^* = b_i, \tag{8.2.22}$$

$$e_j = 0 \quad \text{if and only if} \quad \sum_{i=1}^{n} a_{ij}y_i^* = c_j. \tag{8.2.23}$$

If part: By (8.2.22) we see that

$$\begin{aligned} \sum_{i=1}^{m} b_i y_i^* &= \sum_{i=1}^{m} y_i^* \sum_{j=1}^{n} a_{ij}x_j^* \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} y_i^* a_{ij}x_j^* \end{aligned}$$

In the same way, by using (8.2.23) we see that

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

So, the conclusion is that

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

and the "if part" follows from the Weak Duality Theorem 8.2.12.

Only if part: Like in the proof of the Weak Duality Theorem 8.2.12 we obtain

$$\sum_{j=1}^{n} c_j x_j^* \leq \sum_{i=1}^{m}\sum_{j=1}^{n} y_i^* a_{ij}x_j^* \leq \sum_{i=1}^{m} b_i y_i^*. \tag{8.2.24}$$

Now, by the Strong Duality Theorem 8.2.19, if $\mathbf{x}^*$ and $\mathbf{y}^*$ are optimal, then the LHS of (8.2.24) is equal to the RHS of (8.2.24). But this means that

$$\sum_{j=1}^{n} \left(c_j - \sum_{i=1}^{m} y_i^* a_{ij} \right) x_j^* = 0. \tag{8.2.25}$$

Now, both $\mathbf{x}^*$ and $\mathbf{y}^*$ are feasible. This means that the terms in (8.2.25) are all non-negative. This implies that the terms are all zeros. But this means

that that $e_j x_j = 0$. The validity of $s_i y_i = 0$ can be seen in the same way by considering the equality

$$\sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{m} a_{ij} x_j^* \right) y_i^* \quad = \quad 0.$$

This finishes the proof of the Complementary Slackness Theorem 8.2.20. □

As an example of the use of the Complementary Slackness Theorem 8.2.20, let us consider solving the following LP:

8.2.26 Example. You want to solve the LP

$$\begin{array}{lrcrcrcrcl} \min & w & = & 4y_1 & + & 12y_2 & + & y_3 & & \\ \text{s.t.} & & & y_1 & + & 4y_2 & - & y_3 & \geq & 1 \\ & & & 2y_1 & + & 2y_2 & + & y_3 & \geq & 1 \\ & & & & & & & y_1, y_2, y_3 & \geq & 0 \end{array}$$

Now, suppose you have already solved, e.g. graphically — which is challenging for the LP in 8.2.26 — the much easier LP

$$\begin{array}{lrcrcrcl} \max & z & = & x_1 & + & x_2 & & \\ \text{s.t.} & & & x_1 & + & 2x_2 & \leq & 4 \\ & & & 4x_1 & + & 2x_2 & \leq & 12 \\ & & & -x_1 & + & x_2 & \leq & 1 \\ & & & & & x_1, x_2 & \geq & 0 \end{array}$$

The solution to this dual is

$$\begin{array}{rcl} x_1^* & = & 8/3 \\ x_2^* & = & 2/3 \end{array}$$

with the optimal value

$$\begin{array}{rcl} z^* & = & x_1^* + x_2^* \\ & = & 10/3. \end{array}$$

This means that you have already solved the dual — or primal, if you take the opposite point of view — of the Example 8.2.26.

Now, how can the solution above, combined with the Complementary Slackness Theorem 8.2.20, help you to solve the LP of Example 8.2.26?

Here is how: First note that $x_1^* > 0$ and $x_1^* > 0$. So, the Complementary Slackness Theorem 8.2.20 tells us that the optimal solution $\mathbf{y}^* = [y_1^* \; y_2^* \; y_3^*]'$ of

the LP in Example 8.2.26 must have zero excesses. So, the inequalities in 8.2.26 are actually equalities at the optimum. Also, if we check the optimum $\mathbf{x}^*$ in the first three constraints of the maximum problem, we find equalities in the first two of them, and a strict inequality in the third one. So, the Complementary Slackness Theorem 8.2.20 tells us that $y_3^* = 0$. So, in the optimum $\mathbf{y}^*$ of the LP in Example 8.2.26 we must have

$$\begin{array}{rcrcrcl} y_1^* & + & 4y_2^* & & & = & 1 \\ 2y_1^* & + & 2y_2^* & & & = & 1 \\ & & & & y_3^* & = & 0 \end{array}$$

But this is a very easy system to solve. We obtain

$$\begin{array}{rcl} y_1^* & = & 1/3, \\ y_2^* & = & 1/6, \\ y_3^* & = & 0 \end{array}$$

with the optimal value

$$\begin{array}{rcl} w^* & = & 4y_1^* + 12y_2^* + y_3 \\ & = & 10/3. \end{array}$$

8.3 Primal and Dual Sensitivity

By now it should be clear — although we have not stated it explicitly — what is the connection between sensitivity and duality. Let us be explicit. To put it shortly, it is:

$$\begin{aligned}
\mathbf{x}^*_{\text{primal}} &= \pi_{\text{dual}}, \\
\pi_{\text{primal}} &= \mathbf{y}^*_{\text{dual}}, \\
\mathbf{s}^*_{\text{primal}} &= \mathbf{u}_{\text{dual}}, \\
\mathbf{u}_{\text{primal}} &= \mathbf{e}^*_{\text{dual}}, \\
z^*_{\text{primal}} &= w^*_{\text{dual}}.
\end{aligned}$$

In the above $\mathbf{u}_{\text{primal}}$ and $\mathbf{u}_{\text{dual}}$ denote the vectors of reduced costs in the primal and dual, respectively. Also, $\mathbf{s}^*_{\text{primal}}$ and $\mathbf{e}^*_{\text{dual}}$ denote the slacks and excesses of the primal and dual at optimum, respectively. All the other notations should be clear.

8.3.1 Remark. The equality

$$\pi_{\text{primal}} = \mathbf{y}^*_{\text{dual}}$$

explains the name "shadow prices". Indeed, the dual is a "shadow problem". So, the shadow prices of the constraints at the primal optimum are the prices of the dual variables (that are related to the constraints) at the dual optimum. Sometimes the shadow prices are called the dual prices.

Part III

Applications of Linear Programming

Chapter 9

Data Envelopment Analysis

In this chapter we discuss how to apply LP in the problem of *evaluating the relative efficiency of different units, relative only to themselves*. This is a nice application because of three reasons:

1. it is not at all obvious that LP can be used in this problem,
2. the application gives valuable insight to the LP duality,
3. the application itself is extremely useful.

This chapter is adapted from [2, Ch. 2] and from J.E. Beasley's web notes.

9.1 Graphical Introduction to Data Envelopment Analysis

Data Envelopment Analysis and Decision-Making Units

Data envelopment analysis (DEA), occasionally called *frontier analysis*, was introduced by Charnes, Cooper and Rhodes in 1978 (CCR). DEA is a performance measurement technique which can be used for *evaluating the relative efficiency* of *decision-making units* (DMUs). Here DMU is an abstract term for an entity that transforms inputs into outputs. The term is abstract on purpose: Typically one thinks of DMUs as manufacturers of some goods (outputs) who use some resources (inputs). This way of thinking, while correct, is very narrow-minded: DMU is a much more general concept, and DEA can be applied in very diverse situations. Indeed, basically the DMUs can be pretty much anything. The main restrictions are:

1. the DMUs have the same inputs and outputs,
2. the DMUs' inputs and outputs can be measured numerically.

Indeed, if either one of the points 1. or 2. above fails, it would not make any sense to compare the DMUs *quantitatively*, i.e., with *numbers*.

Examples of DMUs to which DEA has been applied are:

- banks,

- police stations,
- hospitals,
- tax offices,
- prisons,
- military defence bases,
- schools,
- university departments.

9.1.1 Remark. There are two points of DEA that must be emphasized:

1. DEA is a *data oriented*. This means that it will only use the data related to the inputs and outputs of the DMUs under consideration. It does not use any extra theoretical — or practical, or philosophical — knowledge. In this respect, it differs from classical comparison methods where DMUs are compared either to a "representative" DMU or to some "theoretically best" DMU.
2. DEA is an *extreme point method*. It does not compare DMUs to any "representative" or "average" DMU. No, DEA compares the different DMUs to the "best" DMU.

Relative Efficiency

DEA is about comparing the *relative* efficiency of DMUs. *Efficiency* is defined by the following meta-mathematical formula:

$$\text{efficiency} \quad = \quad \frac{\text{outputs}}{\text{inputs}}.$$

There is nothing relative in the definition of efficiency above. Well, not as such. The relativity comes in later.

In the next subsection we will illustrate DEA by means of a small example of Kaupþing Bank branches. Note here that much of what you will see below is a *graphical approach* to DEA. This is very useful if you are attempting to explain DEA to those less technically qualified — such as many you might meet in the management world. There is a mathematical approach to DEA that can be adopted however — this is illustrated later in the following sections.

One Input — One Output

9.1.2 Example. Consider a number of Kaupþing Bank's branches. For each branch we have a single output measure: Number of personal transactions completed per week. Also, we have a single input measure: Number of staff.

The data we have is as follows:

Branch	**Personal transactions**	**Number of staff**
Reykjavík	125	18
Akureyri	44	16
Kópavogur	80	17
Hafnarfjörður	23	11

How then can we compare these branches — or DMUs — and measure their performance using this data? A commonly used method is *ratios*, which means that we will compare *efficiencies*.

For our Kaupþing Bank branch example 9.1.2 we have a single input measure, the number of staff, and a single output measure, the number of personal transactions. Hence the meta-mathematical formula

$$\text{efficiency} \quad = \quad \frac{\text{outputs}}{\text{inputs}} \quad = \quad \frac{\text{output}}{\text{input}}$$

is a well-defined mathematical formula — no metas involved.

We have:

Branch	**Personal transactions per staff member**
Reykjavík	6.94
Akureyri	2.75
Kópavogur	4.71
Hafnarfjörður	2.09

Here we can see that Reykjavík has the highest ratio of personal transactions per staff member, whereas Hafnarfjörður has the lowest ratio of personal transactions per staff member. So, relative to each others, Reykjavík branch is the best (most efficient), and the Hafnarfjörður branch is the worst (least efficient).

As Reykjavík branch is the most efficient branch with the highest ratio of 6.94, it makes sense to compare all the other branches to it. To do this we calculate their *relative efficiency* with respect to Reykjavík branch: We divide

the ratio for any branch by the Reykjavík's efficiency 6.94, and multiply by 100% (which is one) to convert to a percentage. This gives:

Branch	**Relative Efficiency**		
Reykjavík	$100\% \times (6.94/6.94)$	=	100%
Akureyri	$100\% \times (2.75/6.94)$	=	40%
Kópavogur	$100\% \times (4.71/6.94)$	=	68%
Hafnarfjörður	$100\% \times (2.09/6.94)$	=	30%

The other branches do not compare well with Reykjavík. That is, they are *relatively less efficient* at using their given input resource (staff members) to produce output (number of personal transactions).

9.1.3 Remark. We could, if we wish, use the comparison with Reykjavík to set *targets* for the other branches:

1. We could set a target for Hafnarfjörður of continuing to process the same level of output but with one less member of staff. This is an example of an *input target* as it deals with an input measure.
2. An example of an *output target* would be for Hafnarfjörður to increase the number of personal transactions by 10% — e.g. by obtaining new accounts.

We could, of course, also set Hafnarfjörður a mix of input and output targets which we want it to achieve.

One Input — Two Outputs

Typically we have more than one input and one output. In this subsection we consider the case of one input and two outputs. While the case of one input and one output was almost trivial, the case of two outputs and one input is still simple enough to allow for graphical analysis. The analog with LPs would be: LPs with one decision variable are trivial, and LPs with two decision variables are still simple enough to allow for graphical analysis.

Let us extend the Kaupþing Bank branch example 9.1.2:

9.1.4 Example. Consider a number of Kaupþing Bank's branches. For each branch we have a two output measures: Number of personal transactions completed per week, and number of business transaction completed per week. We have a single input measure: Number of staff.

The data we have is as follows:

Branch	**Personal transactions**	**Business transactions**	**Number of staff**
Reykjavík	125	50	18
Akureyri	44	20	16
Kópavogur	80	55	17
Hafnarfjörður	23	12	11

How now can we compare these branches and measure their performance using this data? As before, a commonly used method is ratios, just as in the case considered before of a single output and a single input. Typically we take one of the output measures and divide it by one of the input measures.

For our bank branch example 9.1.4 the input measure is plainly the number of staff (as before) and the two output measures are number of personal transactions and number of business transactions. Hence we have the two ratios:

Branch	**Personal transactions per staff member**	**Business transactions per staff member**
Reykjavík	6.94	2.78
Akureyri	2.75	1.25
Kópavogur	4.71	3.24
Hafnarfjörður	2.09	1.09

Here we can see that Reykjavík has the highest ratio of personal transactions per staff member, whereas Kópavogur has the highest ratio of business transactions per staff member. So, it seems that Reykjavík and Kópavogur are the best performers. Akureyri and Hafnarfjörður do not compare so well with Reykjavík and Kópavogur. That is, they are relatively less efficient at using their given input resource (staff members) to produce outputs (personal and business transactions).

One problem with comparison via ratios is that different ratios can give a different picture and it is difficult to combine the entire set of ratios into a single numeric judgement. For example, consider Akureyri and Hafnarfjörður:

- Akureyri is $2.75/2.09 = 1.32$ times as efficient as Hafnarfjörður at personal transactions,
- Akureyri is $1.25/1.09 = 1.15$ times as efficient as Hafnarfjöður at business transactions.

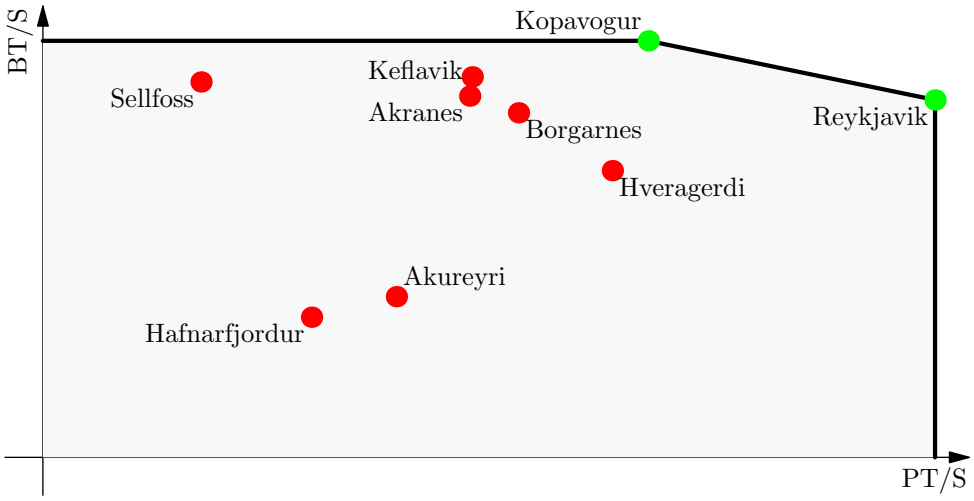
How would you combine these figures — 1.32 and 1.15 — into a single judgement? This problem of different ratios giving different pictures would be especially true if we were to increase the number of branches or increase the number of inputs or outputs.

9.1.5 Example. We ad five extra branches, Sellfoss, Hveragerði, Akranes, Borgarnes, and Keflavík, to Example 9.1.4. The data is now:

Branch	**Personal transactions per staff member**	**Business transactions per staff member**
Reykjavík	6.94	2.78
Akureyri	2.75	1.25
Kópavogur	4.71	3.24
Hafnarfjörður	2.09	1.09
Sellfoss	1.23	2.92
Hveragerði	4.43	2.23
Akranes	3.32	2.81
Borgarnes	3.70	2.68
Keflavík	3.34	2.96

What can be now said about the efficiencies of the branches?

One way around the problem of interpreting different ratios, at least for problems involving just two outputs and a single input, is a simple *graphical analysis*. Suppose we plot the two ratios for each branch as shown below. In the picture we have no ticks to express the scale. The ticks are left out on purpose: DEA is about *relative* efficiency. So, the scales do not matter.



The positions of Reykjavík and Kópavogur in the graph demonstrate that they are superior to all other branches: They are the extreme points, other DMUs

are inferior to them. The line drawn in the picture is called the *efficient frontier*. It was drawn by taking the extreme points, and then connecting them to each others and to the axes. That was a very vague drawing algorithm, but I hope you got the picture.

9.1.6 Remark. The name *Data Envelopment Analysis* arises from the efficient frontier that envelopes, or encloses, all the data we have.

9.1.7 Definition. We say that any DMU on the efficient frontier is 100% *efficient*.

In our Kaupþing Bank branch examples 9.1.4 and 9.1.5, Reykjavík and Kópavogur branches are 100% efficient. This is not to say that the performance of Reykjavík or Kópavogur could not be improved. It may, or may not, be possible to do that. However, we can say that, based on the evidence provided by the different branches, we have no idea of the extent to which their performance can be improved.

9.1.8 Remark. It is important to note here that:

- DEA only gives relative efficiencies, i.e., efficiencies relative to the data considered. It does not — and cannot — give absolute efficiencies.
- No extra information or theory was used in determining the relative efficiencies of the DMUs. What happened was that we merely took data on inputs and outputs of the DMUs we considered, and presented the data in a particular way.
- The statement that a DMU is 100% efficient simply means that we have no other DMU that can be said to be better than it.

Now we know when a DMU is 100% efficient: A DMU is 100% efficient if it is on the efficient frontier. How about the non-efficient DMUs? Can we associate the DMUs that are not in the efficient frontier with a number representing their efficiency? We can. How to do this, is explained below. So, we will now discuss about quantifying efficiency scores for inefficient DMUs.

Let us take Hafnarfjörður as an example of a non-efficient branch. We can see that, with respect to both of the ratios Reykjavík — and Kópavogur, too — dominates Hafnarfjörður. Plainly, Hafnarfjörður is less than 100% efficient. But how much? Now, Hafnarfjörður has

- number of staff 11,
- personal transactions 23,
- personal transactions per staff member $23/11 = 2.09$,
- business transactions 12,
- business transactions per staff member $12/11 = 1.09$.

For Hafnarfjörður we have the ratio

$$\frac{\text{personal transactions}}{\text{business transactions}} = \frac{23}{12} = 1.92.$$

This means that there are 1.92 personal transactions for every business transaction. This figure of 1.92 is also the ratio

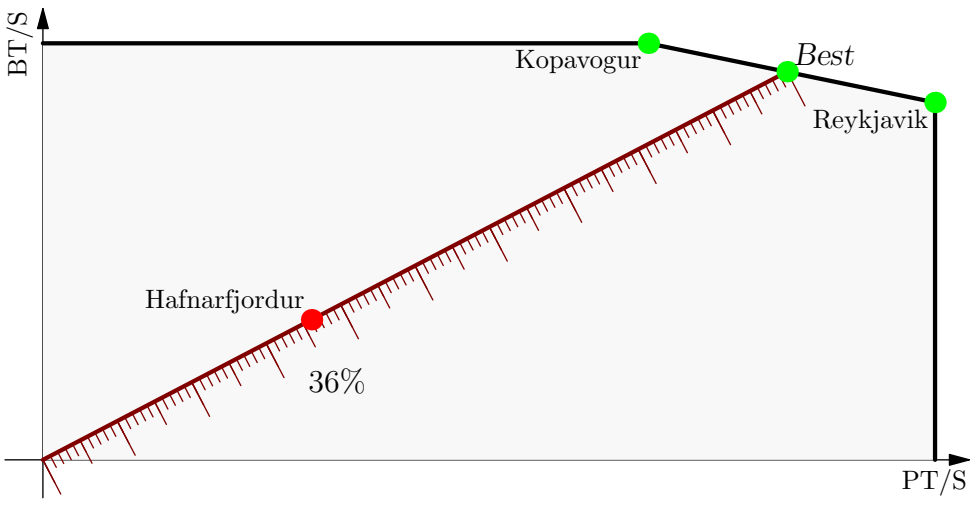
$$\frac{\text{personal transactions per staff member}}{\text{business transactions per staff member}}.$$

Indeed,

$$\begin{aligned}
&\frac{\text{personal transactions}}{\text{business transactions}} \\
&= \frac{\text{personal transactions}}{\text{business transactions}} \times \frac{\text{number of staff members}}{\text{number of staff members}} \\
&= \frac{\text{personal transactions} \times \text{number of staff members}}{\text{business transactions} \times \text{number of staff members}} \\
&= \frac{\text{personal transactions} \,/\, \text{number of staff members}}{\text{business transactions} \,/\, \text{number of staff members}} \\
&= \frac{\text{personal transactions per staff member}}{\text{business transactions per staff member}}.
\end{aligned}$$

This number, 1.92, is the *business mix* of the Hafnarfjörður branch. It can be also be interpreted that Hafnarfjörður branch weighs its outputs, personal transactions and business transactions, so that personal transactions get weight 1.92 and business transactions get weight 1.

Consider now the diagram below. In the diagram we have removed all the inefficient branches, except Hafnarfjörður. The line with the percentage ruler attached drawn through Hafnarfjörður represent all the possible — or virtual, if you like — branches having the same business mix, 1.92, as Hafnarfjörður.



Note the virtual branch *Best* in the picture above. *Best* represents a branch that, were it to exist, would have the same business mix as Hafnarfjörður and would have an efficiency of 100%. Now, since *Best* and Hafnarfjörður have the same business mix, it makes sense to compare them numerically. Here is how to do it: Hafnarfjörður's relative position in the ruler line from the worst branch with the same business mix (the origin) to the best branch with the same business mix (*Best*) is 36%. In other words, 36% of the ruler line is before Hafnarfjörður, and 64% of the ruler line is after Hafnarfjörður. So, it makes sense to say that Hafnarfjörður is, relative to the best branches, 36% efficient — or 64% inefficient, if you like.

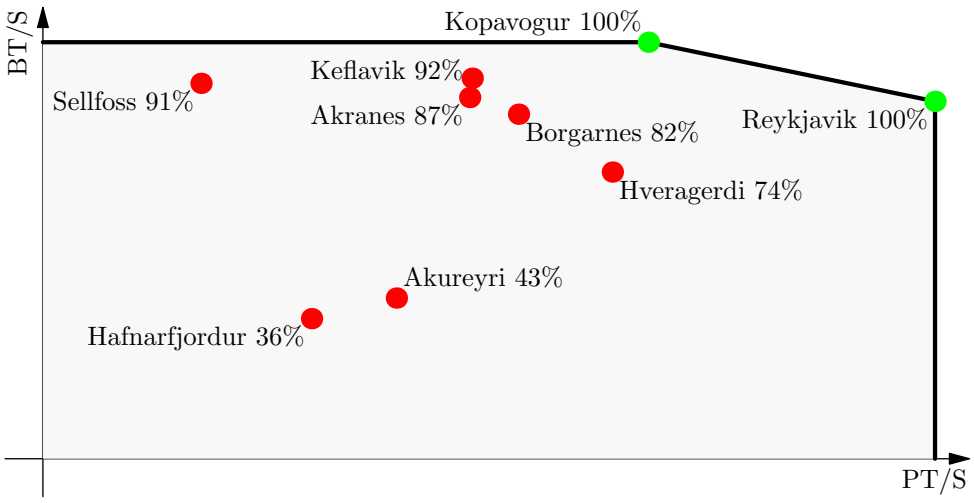
So, given the graphical consideration above we have the following definition for the (relative) efficiency of a DMU with two outputs and one input:

9.1.9 Definition. Draw a line segment from the origin through the DMU in question until you hit the efficient frontier. The *efficiency* of the DMU is

$$\frac{\text{length of the line segment from the origin to the DMU}}{\text{total length of the line segment}} \times 100\%.$$

9.1.10 Remark. The picture — and the definition — above is relative: You can change the scale of either the PT/S or the BT/S axis, or even switch the axes, but the relative efficiency of the Hafnarfjörður branch — or any other branch — won't change.

In the next picture we have written the relative efficiencies of the DMUs (Kaupþing Bank's branches).



The data of the picture above is written in tabular form below. Now it is up to you to decide which one of these two ways of presenting the data you

prefer. The author prefers the picture, and therefore has huge respect for the "tabular-minded".

Branch	**Personal transactions per staff member**	**Business transactions transactions per staff member**	**Relative efficiency**
Reykjavík	6.94	2.78	100%
Akureyri	2.75	1.25	43%
Kópavogur	4.71	3.24	100%
Hafnarfjörður	2.09	1.09	36%
Sellfoss	1.23	2.92	91%
Hveragerði	4.43	2.23	74%
Akranes	3.32	2.81	87%
Borgarnes	3.70	2.68	82%
Keflavik	3.34	2.96	92%

9.1.11 Remark. Consider the picture with the ruler going from the origin through Hafnarfjörður to the efficient frontier. The point labelled *Best* on the efficient frontier is considered to represent the best possible performance that Hafnarfjörður can reasonably be expected to achieve. There are a number of ways by which Hafnarfjörður can move towards that point.

1. It can reduce its input (number of staff) while keeping its outputs (personal and business transaction) constant. This is an input target.
2. It can increase both its outputs, retaining the current business mix ratio of 1.92 while keeping its input (number of staff). This is an output target.
3. It can do some combination of the above.

9.1.12 Remark. It is important to be clear about the appropriate use of the (relative) efficiencies we have calculated. Here we have, e.g.,

- Reykjavík 100%,
- Kópavogur 100%,
- Hafnarfjörður 36%.

This does *not* automatically mean that Hafnarfjörður is only approximately one-third as efficient as the best branches. Rather the efficiencies here would usually be taken as indicative of the fact that other branches are adopting practices and procedures which, if Hafnarfjörður were to adopt them, would enable it to improve its performance. This naturally invokes issues of highlighting and disseminating examples of best practice. Equally there are issues relating to the identification of poor practice.

We end this subsection by further illustrating the relative nature of the DEA efficiencies. We shall ad two extra branches to Example 9.1.5 — Surtsey and Flatey — and see what happens.

Let us start with Surtsey:

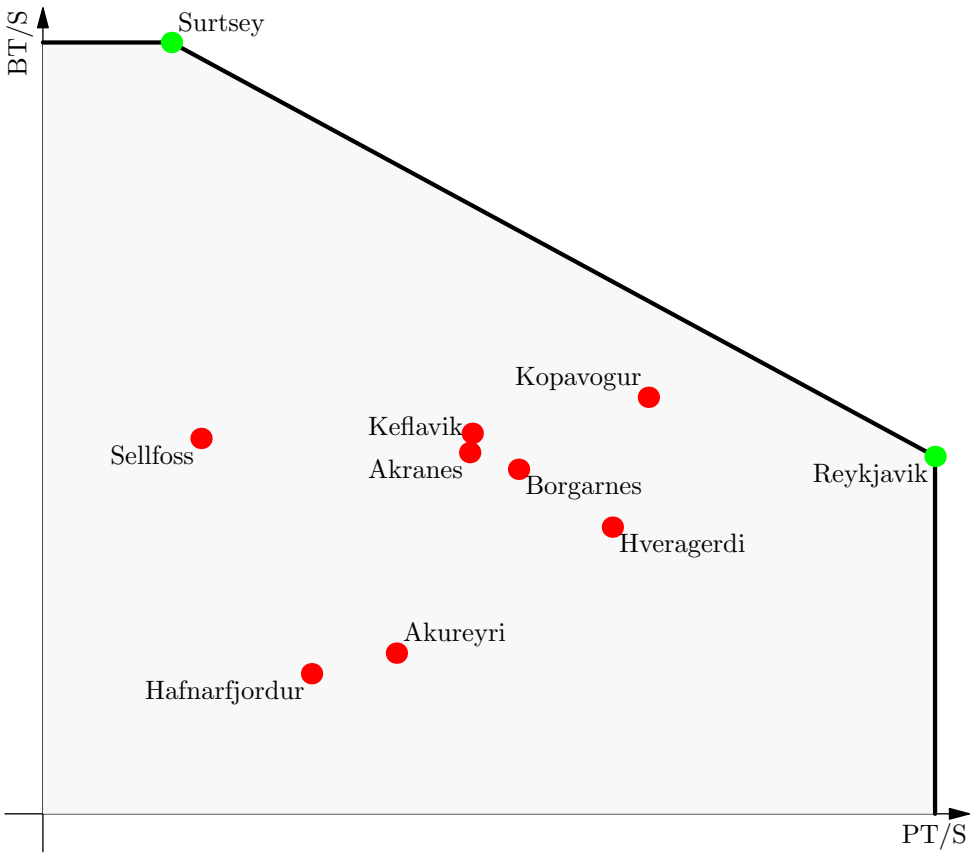
9.1.13 Example. Suppose we have an extra branch, Surtsey, added to the branches of Example 9.1.5. Assume that Surtsey has

1. 1 personal transactions per staff member, and
2. 6 business transactions per staff member.

What changes in the efficiency analysis as a result of including the extra branch Surtsey?

(There are actually no Kaupþing Bank branches in Surtsey. There are no people in Surtsey: People are not allowed in the Fire-Demon's island. There are only puffins is Surtsey.)

The effect of including Surtsey to the graphical Data Envelopment Analysis can be seen in the next picture:



Note that the efficient frontier now excludes Kópavogur. We do not draw that efficient frontier from Reykjavík to Kópavogur and from Kópavogur to Surtsey for two reasons:

1. Mathematically the efficient frontier must be convex,
2. although we have not seen any branches on the line from Reykjavík to Surtsey it is assumed in DEA that we could construct *virtual branches*, which would be a linear combination of Reykjavík and Surtsey, and which would lie on the straight line from Reykjavík to Surtsey.

Also, note that the relative efficiencies of all the inefficient branches have changed. We have not calculated the new relative efficiencies.

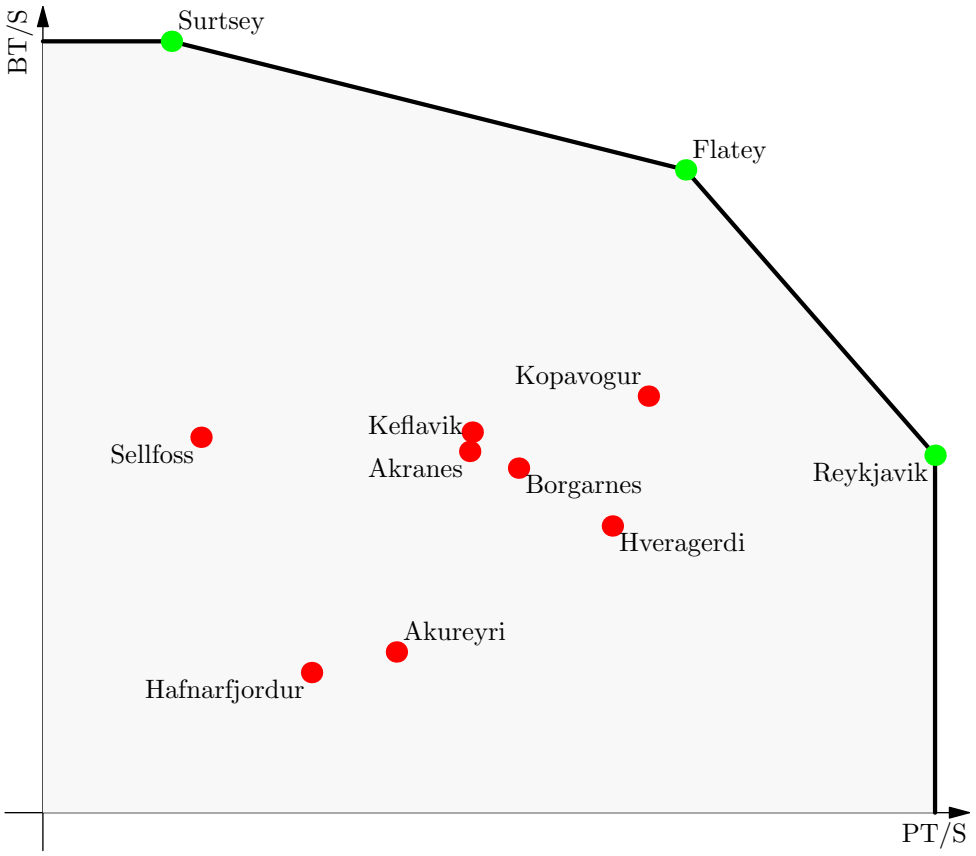
In the above it is clear why Reykjavík and Surtsey have a relative efficiency of 100% (i.e. are efficient): Both are the top performers with respect to one of the two ratios we are considering. The example below, where we have added the Flatey branch, illustrates that a branch can be efficient even if it is not a top performer. In the diagram below Flatey is efficient since under DEA it is judged to have "strength with respect to both ratios", even though it is not the top performer in either.

9.1.14 Example. Suppose we have an extra branch, Flatey, added to the branches of Example 9.1.5. Assume that Flatey has

1. 5 personal transactions per staff member, and
2. 5 business transactions per staff member.

What changes as a result of this extra branch being included in the analysis?

Here is the new picture with Flatey added. Note that Flatey is on the efficient frontier, i.e., 100% efficient, but it is not at top performer in either of the criteria "personal transactions per staff" (PT/S) or "business transactions per staff" (BT/S).



Multiple Input — Multiple Output

In our simple examples 9.1.4, 9.1.5, 9.1.13, and 9.1.14 of the Kaupþing Bank branches we had just one input and two outputs. This is ideal for a simple graphical analysis. If we have more inputs or outputs then drawing simple pictures is not possible without sculptures. However, it is still possible to carry out exactly the same analysis as before, but using mathematics rather than pictures.

In words DEA, in evaluating any number of DMUs, with any number of inputs and outputs:

1. requires the inputs and outputs for each DMU to be specified,
2. defines efficiency for each DMU as a weighted sum of outputs divided by a weighted sum of inputs, where
3. all efficiencies are restricted to lie between zero and one (i.e. between 0% and 100%),
4. in calculating the numerical value for the efficiency of a particular DMU weights are chosen so as to maximize its efficiency, thereby presenting the DMU in the best possible light.

How to carry out the vague four-point list presented above is the topic of the next section 9.2.

9.2 Charnes–Cooper–Rhodes Model

Now we consider mathematically what we have considered graphically in Section 9.1.

We consider n Decision Making Units (DMUs). We call them unimaginatively as DMU_1, DMU_2, DMU_3, and so on upto DMU_n. We are interested in assigning a measure of relative efficiency for each DMU without resorting to any other data than the one provided by the inputs and output of the DMUs themselves.

Data Envelopment Analysis with Matrices

We assume that each DMU has m inputs and s outputs. So, the m inputs of the DMU_k are

$$\mathbf{x}_{\bullet k} = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{mk} \end{bmatrix}.$$

In the same way the s outputs of the DMU_k are

$$\mathbf{y}_{\bullet k} = \begin{bmatrix} y_{1k} \\ \vdots \\ y_{sk} \end{bmatrix}.$$

If we collect the inputs and the outputs into single matrices we have the *input matrix*

$$\mathbf{X} = [x_{jk}] = [\mathbf{x}_{\bullet 1} \cdots \mathbf{x}_{\bullet n}] = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn}, \end{bmatrix}$$

and the *output matrix*

$$\mathbf{Y} = [y_{ik}] = [\mathbf{y}_{\bullet 1} \cdots \mathbf{y}_{\bullet n}] = \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{s1} & \cdots & y_{sn} \end{bmatrix}.$$

So,

$$\begin{aligned} x_{jk} &= \text{the input } j \text{ of the } \mathrm{DMU}_k, \\ y_{ik} &= \text{the output } i \text{ of the } \mathrm{DMU}_k. \end{aligned}$$

Charnes–Cooper–Rhodes Fractional Program

Given what we have learnt it seems reasonable to measure the (relative) efficiency of the DMUs as weighted sums. So, let

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_s \end{bmatrix}$$

be the weights associated with the s outputs of the DMUs. Similarly, let

$$\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}$$

be the weights associated with the inputs of the DMUs. Then the *weighted efficiency*, with weights $\mathbf{u}$ and $\mathbf{v}$, of any DMU, say DMU_o (o for DMU under *O*bservation) is

$$\begin{aligned} h_o(\mathbf{u},\mathbf{v}) &= \text{the } (\mathbf{u},\mathbf{v}) \text{ weighted efficiency of } \text{DMU}_o \\ &= \frac{\mathbf{u} \text{ weighted outputs of } \text{DMU}_o}{\mathbf{v} \text{ weighted inputs of } \text{DMU}_o} \\ &= \frac{\sum_{j=1}^{s} u_j y_{jo}}{\sum_{i=1}^{m} v_i x_{io}} \\ &= \frac{\mathbf{u}'\mathbf{y}_{\bullet o}}{\mathbf{v}'\mathbf{x}_{\bullet o}}. \end{aligned} \tag{9.2.1}$$

9.2.2 Example. Consider the Kaupþing Bank's branches of Example 9.1.4 of the previous section:

Branch	Personal transactions	Business transactions	Number of staff
Reykjavík	125	50	18
Akureyri	44	20	16
Kópavogur	80	55	17
Hafnarfjörður	23	12	11

Denote the data of Example 9.2.2 by

$$\begin{aligned} \mathbf{x}_{1\bullet} &= \text{number of staff,} \\ \mathbf{y}_{1\bullet} &= \text{number of personal transactions,} \\ \mathbf{y}_{2\bullet} &= \text{number of business transactions.} \end{aligned}$$

So, e.g., $\mathbf{x}_{1\bullet}$ is the 4-dimensional row vector consisting of the number of staff data for the DMUs Reykjavík, Akureyri, Kópavogur, and Hafnarfjörður. Similarly, $\mathbf{y}_{1\bullet}$ and $\mathbf{y}_{2\bullet}$ are the 4-dimensional row vectors indicating the number of personal and business transactions for each of the four DMUs: Reykjavík (1), Akureyri (2), Kópavogur (3), and Hafnarfjörður (4). The output matrix for this example is:

$$\begin{aligned}\mathbf{Y} &= \begin{bmatrix} \mathbf{y}_{1\bullet} \\ \mathbf{y}_{2\bullet} \end{bmatrix} \\ &= \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \end{bmatrix} \\ &= \begin{bmatrix} 125 & 44 & 80 & 23 \\ 50 & 20 & 55 & 12 \end{bmatrix}.\end{aligned}$$

The input matrix is

$$\begin{aligned}\mathbf{X} &= \mathbf{x}_{1\bullet} \\ &= [x_{11}\ x_{12}\ x_{13}\ x_{14}] \\ &= [18\ 16\ 17\ 11].\end{aligned}$$

Let us then take the Hafnarfjörður branch under consideration. So, $\mathrm{DMU}_o = \mathrm{DMU}_4$ is Hafnarfjörður. With our vector notation Hafnarfjörður would have the (weighted) efficiency

$$h_o(\mathbf{u},\mathbf{v}) = \frac{u_1 y_{1o} + u_2 y_{2o}}{v_1 x_{1o}} = \frac{u_1 \times 23 + u_2 \times 12}{v_1 \times 11}.$$

Now there is the problem of fixing the weight $\mathbf{u}$ and $\mathbf{v}$ of the outputs and the inputs. Each DMU would — of course — want to fix the weights $\mathbf{u}$ and $\mathbf{v}$ in such a way that they would look best in comparison with the other DMUs. So, it is in the interests of each and every one of the DMUs to maximize the weighted efficiency $h_o(\mathbf{u},\mathbf{v})$. In particular, this means that Hafnarfjörður faces an optimization problem

$$\max_{\mathbf{u},\mathbf{v}} h_o(\mathbf{u},\mathbf{v}) = \max_{u_1,u_2,v_1} \frac{u_1 \times 23 + u_2 \times 12}{v_1 \times 11}. \tag{9.2.3}$$

Obviously there must be constraints to the decision variables $\mathbf{u}$ and $\mathbf{v}$. Indeed, otherwise the optimization problem (9.2.3) would yield an unbounded optimum. So, what are the constraints? Well, obviously we have the sign constraints

$$\mathbf{u},\mathbf{v} \geq \mathbf{0}.$$

This does not help too much yet, though. The optimum of (9.2.3) is still unbounded. Now, we remember that we are dealing with efficiencies. But, an efficiency is a number between 0% and 100%. So, we have the constraint

$$h_o(\mathbf{u},\mathbf{v}) \leq 1.$$

This does not help too much either. Indeed, now the optimum for (9.2.3) would be 1, or 100%. But we are close now. Remember that the efficiency is *always* a number between 0% and 100%. So, the efficiencies of the other DMUs must also be between 0% and 100%. So, we let Hafnarfjörður set the weights $\mathbf{u}$ and $\mathbf{v}$, and the other DMUs are then measured in the same way. So, the constraints are

$$h_k(\mathbf{u}, \mathbf{v}) \quad \leq \quad 1 \qquad \text{for all DMU}_k,\ k = 1, \ldots, n.$$

Collecting what we have found above we have found the *fractional form* of the Charnes–Cooper–Rhodes (CCR) model

9.2.4 Definition. The CCR Fractional Program for DMU_o relative to $\text{DMU}_1, \ldots, \text{DMU}_n$ is

$$\begin{aligned} \max \theta &= \frac{\mathbf{u}'\mathbf{y}_{\bullet o}}{\mathbf{v}'\mathbf{x}_{\bullet o}} \\ \text{s.t.} \qquad \frac{\mathbf{u}'\mathbf{y}_{\bullet k}}{\mathbf{v}'\mathbf{x}_{\bullet k}} &\leq 1 \qquad \text{for all } k = 1, \ldots, n \\ \mathbf{u}, \mathbf{v} &\geq 0. \end{aligned} \tag{9.2.5}$$

The figure θ is the DMU_0's *DEA Efficiency*.

Charnes–Cooper–Rhodes Linear Program

Consider the optimization problem in Definition 9.2.4. This is not an LP. But the name of this part of the Chapters is "Applications of Linear Programming". So, it seems that we have a misnomer! Also, we do not know how to solve fractional programs like the (9.2.5) in Definition 9.2.4. Fortunately there is a way of transforming the fractional program (9.2.5) into an LP.

Before going to the LP let us note that while the efficiency score θ of the CCR fractional program (9.2.5) is unique, there are many different weights $\mathbf{u}, \mathbf{v}$ that give the same efficiency. Indeed, if the weights $\mathbf{u}, \mathbf{v}$ give the optimal efficiency, then so do the weights $\alpha\mathbf{u}, \alpha\mathbf{v}$ for any $\alpha > 0$. This is due to the fact that we are dealing with ratios. Indeed, for any $\alpha > 0$

$$\begin{aligned} h_o(\mathbf{u}, \mathbf{v}) &= \frac{\mathbf{u}'\mathbf{y}_{\bullet o}}{\mathbf{v}'\mathbf{x}_{\bullet o}} \\ &= \frac{\alpha\mathbf{u}'\mathbf{y}_{\bullet o}}{\alpha\mathbf{v}'\mathbf{x}_{\bullet o}} \\ &= h_o(\alpha\mathbf{u}, \alpha\mathbf{v}). \end{aligned}$$

There is an easy way out, however. We just normalize the denominator in the ratio, i.e., we insist that

$$\mathbf{v}'\mathbf{x}_{\bullet o} \quad = \quad 1.$$

Now we are ready to give the LP formulation of the fractional program 9.2.5:

9.2.6 Definition. The *CCR LP* for DMU_o relative to $\mathrm{DMU}_1, \ldots, \mathrm{DMU}_n$ is

$$\begin{aligned} \max \theta &= \mathbf{u}'\mathbf{y}_{\bullet o} \\ \text{s.t.} \quad & \mathbf{v}'\mathbf{x}_{\bullet o} = 1, \\ & \mathbf{u}'\mathbf{Y} \leq \mathbf{v}'\mathbf{X} \\ & \mathbf{u}, \mathbf{v} \geq 0. \end{aligned} \tag{9.2.7}$$

The figure θ is the DMU_o's *DEA Efficiency*.

It may not be immediately clear why the LP (9.2.7) is the same optimization problem as the fractional program (9.2.5). So, we explain a little why this is so. Consider the fractional program (9.2.5). First, note that the extra assumption $\mathbf{v}'\mathbf{x}_{\bullet o} = 1$ does not change the optimal value of θ in the fractional program. Indeed, we have already seen that this restriction merely chooses one optimal choice among many. Next note that in the LP (9.2.7) we have

$$\theta \quad = \quad \mathbf{u}'\mathbf{y}_{\bullet o},$$

while in the fractional program (9.2.5) we have

$$\theta \quad = \quad \frac{\mathbf{u}'\mathbf{y}_{\bullet o}}{\mathbf{v}'\mathbf{x}_{\bullet o}}.$$

Remember, that we have now the normalizing assumption $\mathbf{v}'\mathbf{x}_0 = 1$. So, we see that the fractional and linear objectives are actually the same. Finally, let us look the constraints

$$\frac{\mathbf{u}'\mathbf{y}_{\bullet k}}{\mathbf{v}'\mathbf{x}_{\bullet k}} \quad \leq \quad 1$$

of the fractional program (9.2.5) and compare them to the constraints

$$\mathbf{u}'\mathbf{y}_{\bullet k} \quad \leq \quad \mathbf{v}'\mathbf{x}_{\bullet k}$$

of the linear program (9.2.7) (written in the matrix form there). If you multiply both sides of the fractional programs constraints by $\mathbf{v}'\mathbf{x}_{\bullet k}$ you notice that these constraints are actually the same. So, we see that the fractional program (9.2.5) and the linear program (9.2.7) are the same.

Efficiency for Hafnarfjörður and Reykjavík

Let us calculate mathematically, as opposed to graphically, Hafnarfjörður's efficiency and Reykjavík's efficiency in Example 9.2.2 by using the CCR LP (9.2.7).

Recall the data

Branch	**Personal transactions**	**Business transactions**	**Number of staff**
Reykjavík	125	50	18
Akureyri	44	20	16
Kópavogur	80	55	17
Hafnarfjörður	23	12	11

and the notation

$$\begin{aligned} \mathbf{x}_{1\bullet} &= \text{number of staff,} \\ \mathbf{y}_{1\bullet} &= \text{number of personal transactions,} \\ \mathbf{y}_{2\bullet} &= \text{number of business transactions.} \end{aligned}$$

9.2.8 Remark. Note that $\mathbf{x}_{1\bullet}, \mathbf{y}_{1\bullet}, \mathbf{y}_{2\bullet}$ are *not* the decision variables. They are the data. The decision variables are the weights v_1, u_1, u_2 associated with the data $\mathbf{x}_{1\bullet}, \mathbf{y}_{1\bullet}, \mathbf{y}_{2\bullet}$.

Here is the LP for Hafnarfjörður

$$\begin{array}{lrcrcll} \max \theta & = & 23u_1 & + & 12u_2 & & & \text{(DEA Efficiency)} \\ \text{s.t.} & & 11v_1 & & & = & 1 & \text{(Normalization)} \\ & & 125u_1 & + & 50u_2 & \leq & 18v_1 & \text{(DMU Reykjavik)} \\ & & 44u_1 & + & 20u_2 & \leq & 16v_1 & \text{(DMU Akureyri)} \\ & & 80u_1 & + & 55u_2 & \leq & 17v_1 & \text{(DMU Kopavogur)} \\ & & 23u_1 & + & 12u_2 & \leq & 11v_1 & \text{(DMU Hafnarfjordur)} \\ & & & & u_1, u_2, v_1 & \geq & 0 & \end{array}$$

Now, this LP is certainly not in standard form. We shall solve it with GLPK, however. So, there is no reason to transform it into a standard form.

Here is the GNU MathProg code for the Hafnarfjörður's LP:

```
# DEA efficiency for Hafnarfjordur

# Decision variables

var u1 >=0; # weight for personal transactions
var u2 >=0; # weight for business transactions
var v1 >=0; # weight for staff members

# Hafnarfjordur's DEA efficiency
maximize theta: 23*u1 + 12*u2;

# normalization constraint
s.t. Norm: 11*v1 = 1;

#  constraints from the set of DMUs
s.t. Reykjavik:     125*u1+50*u2 <= 18*v1;
s.t. Akureyri:       44*u1+20*u2 <= 16*v1;
s.t. Kopavogur:      80*u1+55*u2 <= 17*v1;
s.t. Hafnarfjordur:  23*u1+12*u2 <= 11*v1;

end;
```

Here is the "Simplex section" of the `glpsol` report:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 theta        B       0.361739
     2 Norm         NS             1             1             =      0.361739
     3 Reykjavik    NU             0                          -0      0.106087
     4 Akureyri     B      -0.826561                          -0
     5 Kopavogur    NU             0                          -0      0.121739
     6 Hafnarfjordur
                    B      -0.638261                          -0

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 u1           B     0.00442688             0
     2 u2           B      0.0216601             0
     3 v1           B      0.0909091             0
```

We see that the DEA efficiency of Hafnarfjörður is 36%. This is no news to us. We learnt this in the previous section with graphical analysis. But it is nice to see that the graphical and the mathematical approach agree of the efficiency.

Let us then consider the Reykjavík branch in Example 9.2.2. Here is the GNU MathProg code for the Reykjavík branch:

```
# DEA efficiency for Reykjavik

# Decision variables

var u1 >=0; # weight for personal transactions
var u2 >=0; # weight for business transactions
var v1 >=0; # weight for staff members

# Reykjavik's DEA efficiency
maximize theta: 125*u1 + 50*u2;

# normalization constraint
s.t. Norm: 18*v1 = 1;

#  constraints from the set of DMUs
s.t. Reykjavik:     125*u1+50*u2 <= 18*v1;
s.t. Akureyri:       44*u1+20*u2 <= 16*v1;
s.t. Kopavogur:      80*u1+55*u2 <= 17*v1;
s.t. Hafnarfjordur:  23*u1+12*u2 <= 11*v1;

end;
```

And here is the "Simplex part" of the `glpsol` report:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 theta        B              1
     2 Norm         NS             1             1             =             1
     3 Reykjavik    NU             0                          -0             1
     4 Akureyri     B      -0.536889                          -0
     5 Kopavogur    B      -0.304444                          -0
     6 Hafnarfjordur
                    B      -0.427111                          -0

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 u1           B          0.008             0
     2 u2           NL             0             0                       < eps
     3 v1           B      0.0555556             0
```

We see that the mathematical approach agrees with the graphical approach: Reykjavík is 100% efficient.

9.3 Charnes–Cooper–Rhodes Model's Dual

Finding the DEA efficiency of a DMU_o in the CCR model is an LP (9.2.7). So, it must have a dual LP associated with it. In this section we explain how to construct the dual, and how to interpret it. Also, in the last subsection we illustrate how to find the DEA efficiencies of the Hafnarfjörður and Reykjavík branch of Example 9.2.2 by using the the dual of the CCR model.

Finding Dual

In this subsection we will find the dual of the CCR LP (9.2.7) by using brute force matrix calculus. We throw all intuition to the wind, and simply follow mathematical algorithms. In the next subsection we shall interpret the dual we find in this subsection.

Recall the LP form of the CCR model:

$$
\begin{aligned}
&\max \theta = \mathbf{u}'\mathbf{y}_{\bullet o} \\
&\text{s.t.} \quad \mathbf{v}'\mathbf{x}_{\bullet o} = 1, \\
&\qquad \mathbf{u}'\mathbf{Y} \leq \mathbf{v}'\mathbf{X} \\
&\qquad \mathbf{u}, \mathbf{v} \geq 0.
\end{aligned}
\tag{9.3.1}
$$

To find the dual, we write the LP (9.3.1) in standard form. We could use the block matrix notation, but the derivation is probably easier to understand if we do not use the concise matrix notation. So, we abandon matrices in the derivation.

Without matrices the CCR LP (9.3.1) can be written as

$$
\begin{array}{lrcl}
\max\ \theta\ = & u_1 y_{1o} + \cdots + u_s y_{so} & & \\
\text{s.t.} & v_1 x_{1o} + \cdots + v_m x_{mo} & = & 1 \\
& u_1 y_{11} + \cdots + u_s y_{s1} & \leq & v_1 x_{11} + \cdots + v_m x_{mo} \\
& \vdots & \vdots & \vdots \\
& u_1 y_{1n} + \cdots + u_s y_{sn} & \leq & v_1 x_{1n} + \cdots + v_m x_{mn} \\
& u_1, \ldots, u_s, v_1, \ldots, v_m & \geq & 0.
\end{array}
\tag{9.3.2}
$$

The LP (9.3.2) is certainly not in standard form. Actually, it is pretty far from it. As a first step in transforming (9.3.2) into a standard form let us write it so that the decision variables $\mathbf{u} = [u_1 \ \cdots \ u_s]'$ and $\mathbf{v} = [v_1 \ \cdots \ v_m]'$ are in the right places, i.e., coefficients are in front of the decision variables, all the decision variables are represented everywhere, and there are no decision

variables in the RHSs. We obtain:

(9.3.3)

$$\begin{array}{lrcrcrcrcl}
\max\theta = & y_{1o}u_1 & + \cdots + & y_{so}u_s & +0v_1 & + \cdots + & 0v_m & & \\
\text{s.t.} & 0u_1 & + \cdots + & 0u_s & +x_{1o}v_1 & + \cdots + & x_{mo}v_m & = & 1 \\
& y_{11}u_1 & + \cdots + & y_{s1}u_s & -x_{11}v_1 & - \cdots - & x_{m1}v_m & \leq & 0 \\
& \vdots & & \vdots & & \vdots & \vdots & \vdots & \vdots \\
& y_{1n}u_1 & + \cdots + & y_{sn}u_s & -x_{1n}v_1 & - \cdots - & x_{mn}v_m & \leq & 0 \\
& u_1 & \ldots & u_s & v_1 & \ldots & v_m & \geq & 0
\end{array}$$

Next, we split the equality constraint in (9.3.3) into two $\leq$ inequalities. We obtain:

(9.3.4)

$$\begin{array}{lrcrcrcrcl}
\max\theta = & y_{1o}u_1 & + \cdots + & y_{so}u_s & +0v_1 & + \cdots + & 0v_m & & \\
\text{s.t.} & 0u_1 & + \cdots + & 0u_s & +x_{1o}v_1 & + \cdots + & x_{mo}v_m & \leq & 1 \\
& -0u_1 & - \cdots - & -0u_s & -x_{1o}v_1 & - \cdots - & -x_{mo}v_m & \leq & -1 \\
& y_{11}u_1 & + \cdots + & y_{s1}u_s & -x_{11}v_1 & - \cdots - & x_{m1}v_m & \leq & 0 \\
& \vdots & & \vdots & & \vdots & \vdots & \vdots & \vdots \\
& y_{1n}u_1 & + \cdots + & y_{sn}u_s & -x_{1n}v_1 & - \cdots - & x_{mn}v_m & \leq & 0 \\
& u_1 & \ldots & u_s & v_1 & \ldots & v_m & \geq & 0
\end{array}$$

Now it is pretty straightforward to transform the LP (9.3.3) into the dual. Let ϑ be the objective, and let $\mu = [\mu_1\ \mu_2\mu_3\ \cdots \mu_{n+2}]'$ be the decision variables. We obtain:

(9.3.5)

$$\begin{array}{lrcrcrcl}
\min\vartheta = & \mu_1 - \mu_2 & & & & & & \\
\text{s.t.} & 0\mu_1 - 0\mu_2 & +y_{11}\mu_3 & + \cdots + & y_{1n}\mu_{n+2} & \geq & y_{1o} \\
& \vdots & & \vdots & & \vdots & \vdots \\
& 0\mu_1 - 0\mu_2 & +y_{s1}\mu_3 & + \cdots + & y_{sn}\mu_{n+2} & \geq & y_{so} \\
& x_{1o}\mu_1 - x_{1o}\mu_2 & -x_{11}\mu_3 & - \cdots - & x_{1n}\mu_{n+1} & \geq & 0 \\
& \vdots & & \vdots & & \vdots & \vdots \\
& x_{mo}\mu_1 - x_{mo}\mu_2 & -x_{m1}\mu_3 & - \cdots - & x_{mn}\mu_{n+1} & \geq & 0 \\
& & & & \mu_1, \ldots, \mu_{n+2} & \geq & 0
\end{array}$$

We have found the dual (9.3.5) of the CCR LP (9.2.7). Unfortunately, this dual is not easy to interpret. So, we have to transform it slightly in order to understand what is going on. This is what we do in the next subsection.

Interpreting Dual

Let us substitute the objective

$$\vartheta \quad = \quad \mu_1 - \mu_2$$

into the constraints of (9.3.5). In doing so, we actually eliminate all the occurrences on μ_1 and μ_2 in the system. We obtain:

$$
\begin{array}{lrrrcr}
\min \vartheta & & & & & \\
\text{s.t.} & & y_{11}\mu_3 & +\cdots+ \quad y_{1n}\mu_{n+2} & \geq & y_{1o} \\
 & & & \vdots & \vdots & \vdots \\
 & & y_{s1}\mu_3 & +\cdots+ \quad y_{sn}\mu_{n+2} & \geq & y_{so} \\
 & x_{1o}\vartheta & -x_{11}\mu_3 & -\cdots- \quad x_{1n}\mu_{n+1} & \geq & 0 \\
 & \vdots & & \vdots & \vdots & \vdots \\
 & x_{mo}\vartheta & -x_{m1}\mu_3 & -\cdots- \quad x_{mn}\mu_{n+1} & \geq & 0 \\
 & & & \mu_1, \ldots, \mu_{n+2} & \geq & 0
\end{array}
\tag{9.3.6}
$$

Next, we shall renumber the remaining decision variables. The new decision variables will be $\lambda = [\lambda_1 \ \cdots \ \lambda_n]'$, where $\lambda_1 = \mu_3$, $\lambda_2 = \mu_4$, $\ldots$, $\lambda_n = \mu_{n+2}$. So, the LP (9.3.6) becomes

$$
\begin{array}{lrrrcr}
\min \vartheta & & & & & \\
\text{s.t.} & & +y_{11}\lambda_1 & +\cdots+ \quad y_{1n}\lambda_n & \geq & y_{1o} \\
 & & & \vdots & \vdots & \vdots \\
 & & +y_{s1}\lambda_1 & +\cdots+ \quad y_{sn}\lambda_n & \geq & y_{so} \\
 & x_{1o}\vartheta & -x_{11}\lambda_1 & -\cdots- \quad x_{1n}\lambda_n & \geq & 0 \\
 & \vdots & & \vdots & \vdots & \vdots \\
 & x_{mo}\vartheta & -x_{m1}\lambda_1 & -\cdots- \quad x_{mn}\lambda_n & \geq & 0 \\
 & & & \lambda_1, \ldots, \lambda_n & \geq & 0
\end{array}
\tag{9.3.7}
$$

Finally, we reorganize the $\geq$ inequalities, for a reason that will become apparent later when we get to the interpretation. We obtain:

$$
\begin{array}{lrrcr}
\min \vartheta & & & & \\
\text{s.t.} & y_{11}\lambda_1 & +\cdots+ \quad y_{1n}\lambda_n & \geq & y_{1o} \\
 & & \vdots & \vdots & \vdots \\
 & y_{s1}\lambda_1 & +\cdots+ \quad y_{sn}\lambda_n & \geq & y_{so} \\
 & x_{11}\lambda_1 & +\cdots+ \quad x_{1n}\lambda_n & \leq & x_{1o}\vartheta \\
 & & \vdots & \vdots & \vdots \\
 & x_{m1}\lambda_1 & +\cdots+ \quad x_{mn}\lambda_n & \leq & x_{mo}\vartheta \\
 & & \lambda_1, \ldots, \lambda_n & \geq & 0
\end{array}
\tag{9.3.8}
$$

We have found out a formulation of the dual of the CCR LP (9.2.7) that we can understand in a meaningful way: The dual variables

$$
\lambda \ = \ \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix}
$$

are the weights for a *virtual* DMU — denoted by $\mathrm{DMU}_{\mathrm{virtual}}$ — that will be the reference point of the DMU_o — the DMU under observation. The virtual DMU, $\mathrm{DMU}_{\mathrm{virtual}}$, is constructed from the actual DMUs — $\mathrm{DMU}_1, \ldots, \mathrm{DMU}_n$ — by weighting the actual DMU_k with weight λ_k:

$$\mathrm{DMU}_{\mathrm{virtual}} \quad = \quad \sum_{k=1}^{n} \lambda_k \mathrm{DMU}_k.$$

So, the the restrictions

$$\begin{array}{rcrcl} y_{11}\lambda_1 & + \cdots + & y_{1n}\lambda_n & \geq & y_{1o} \\ \vdots & & \vdots & \vdots & \vdots \\ y_{s1}\lambda_1 & + \cdots + & y_{sn}\lambda_n & \geq & y_{so} \end{array}$$

say

> *All the outputs of the virtual DMU are at least as great as the corresponding outputs of the DMU under observation.*

The restrictions

$$\begin{array}{rcrcl} x_{11}\lambda_1 & + \cdots + & x_{1n}\lambda_n & \leq & x_{1o}\vartheta \\ \vdots & & \vdots & \vdots & \vdots \\ x_{m1}\lambda_1 & + \cdots + & x_{mn}\lambda_n & \leq & x_{mo}\vartheta \end{array}$$

say

> *If the inputs of the DMU under observation are scaled down by ϑ, then all the inputs are at least as great as the corresponding inputs of the virtual DMU.*

Finally, here is the CCR dual LP (9.3.8) in matrix form:

9.3.9 Definition. The *CCR Dual LP* for DMU_o relative to $\mathrm{DMU}_1, \ldots, \mathrm{DMU}_n$ is

$$\begin{array}{ll} & \min \vartheta \\ \text{s.t.} & \vartheta \mathbf{x}_{\bullet o} \geq \mathbf{X}\lambda, \\ & \mathbf{Y}\lambda \geq \mathbf{y}_{\bullet o} \\ & \lambda \geq 0. \end{array} \tag{9.3.10}$$

The figure ϑ is the DMU_o's *DEA Efficiency.*

Dual Efficiency for Hafnarfjörður and Reykjavík

Let us see what are the (dual) DEA efficiencies for the Hafnarfjörður and Reykjavík in Example 9.2.2. We have already found out the solutions in previous sections by using the graphical approach and the primal CCR approach: Hafnarfjörður is 36% DEA efficient and Reykjavík is 100% DEA efficient. So, we shall now check if this third approach — the dual CCR approach — will give the same results, as it should.

Here is the GNU MathProg code for the dual CCR LP (9.3.10) for Hafnarfjörður. Note that we have the objective ϑ as a variable. This is due to the fact that ϑ does not have an equation in the dual CCR LP (9.3.10). The solution of declaring ϑ as a decision variable and then equating it with the objective is not an elegant one, but it works.

```
# Dual DEA efficiency for Hafnarfjordur

# Decision variables

var lambda1 >=0; # weight for Reykjavik
var lambda2 >=0; # weight for Akureyri
var lambda3 >=0; # weight for Kopavogur
var lambda4 >=0; # weight for Hafnarfjordur
var theta; # objective has no equation, so it is a variable

# Hafnarfjordur's dual DEA efficiency
minimize obj: theta;

# Input constraints
s.t.  input1: 18*lambda1 + 16*lambda2 + 17*lambda3 + 11*lambda4
              <= theta*11;  # number of staff

# Output constraints
s.t.  output1: 125*lambda1 + 44*lambda2 + 80*lambda3 + 23*lambda4
               >= 23;  # personal transactions
s.t.  output2: 50*lambda1 + 20*lambda2 + 55*lambda3 + 12*lambda4
               >= 12;  # business transactions

end;
```

Here is the relevant part of the `glpsol` report:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 obj          B       0.361739
     2 input1       NU             0                          -0    -0.0909091
     3 output1      NL            23            23                 0.00442688
     4 output2      NL            12            12                  0.0216601

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 lambda1      B       0.106087             0
     2 lambda2      NL             0             0                  0.826561
     3 lambda3      B       0.121739             0
     4 lambda4      NL             0             0                  0.638261
     5 theta        B       0.361739
```

We see that the (dual) DEA efficiency for Hafnarfjörður is 36%, as it should. We can also read the composition of the virtual DMU associated with Hafnarfjörður:

$$\begin{aligned} \mathrm{DMU}_{\mathrm{virtual}} &= \lambda_1 \mathrm{DMU}_1 + \lambda_2 \mathrm{DMU}_2 + \lambda_3 \mathrm{DMU}_3 + \lambda_4 \mathrm{DMU}_4 \\ &= 10.6\% \times \mathrm{DMU}_{\mathrm{Reykjavik}} + 12.2\% \times \mathrm{DMU}_{\mathrm{Kopavogur}}. \end{aligned}$$

So, one way to interpret the result is:

> *Consider the virtual DMU that is composed of* 10.6% *of Reykjavík and* 12.2% *of Kópavogur. Then the outputs of this virtual DMU are the same as those of Hafnarfjörður, but the virtual DMU uses only* 36% *of the inputs Hafnarfjörður uses.*

Here is the GNU MathProg code for Reykjavík:

```
# Dual DEA efficiency for Reykjavik

# Decision variables

var lambda1 >=0; # weight for Reykjavik
var lambda2 >=0; # weight for Akureyri
var lambda3 >=0; # weight for Kopavogur
var lambda4 >=0; # weight for Hafnarfjordur
var theta; # objective has no equation, so it is a variable

# Reykjavik's dual DEA efficiency
minimize obj: theta;

# Input constraints
s.t.  input1: 18*lambda1 + 16*lambda2 + 17*lambda3 + 11*lambda4
              <= theta*18;  # number of staff

# Output constraints
s.t.  output1: 125*lambda1 + 44*lambda2 + 80*lambda3 + 23*lambda4
               >= 125;  # personal transactions
s.t.  output2: 50*lambda1 + 20*lambda2 + 55*lambda3 + 12*lambda4
               >= 50;  # business transactions

end;
```

Here is the relevant part of the `glpsol` report:

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 obj          B              1
     2 input1       NU             0                          -0    -0.0555556
     3 output1      NL           125           125                       0.008
     4 output2      B             50            50

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 lambda1      B              1             0
     2 lambda2      NL             0             0                  0.536889
     3 lambda3      NL             0             0                  0.304444
     4 lambda4      NL             0             0                  0.427111
     5 theta        B              1
```

We see that Reykjavík is 100% (dual) DEA efficient. We also see that the virtual DMU associated with Reykjavík is Reykjavík itself. This is not surprising: Remember that the virtual DMU was a DMU that "is the same in outputs" but "uses less or equal inputs". Since Reykjavík is 100% efficient, there should not be a virtual DMU that uses less inputs and produces the same outputs.

9.4 Strengths and Weaknesses of Data Envelopment Analysis

Data Envelopment Analysis is a very general framework that draws conclusions from the data available and makes very few — is any — assumptions of the context where the data came from. This is its main strength and this its main weakness.

Strengths

1. DEA is simple enough to be modelled with LPs.
2. DEA can handle multiple input and multiple outputs.
3. DEA does not require an assumption of a functional form relating inputs to outputs. In particular, one does not have to think that the outputs are consequences of the inputs.
4. DMUs are directly compared against a peer or (virtual) combination of peers.
5. Inputs and outputs can have very different units. For example, output y_1 could be in units of lives saved and input x_1 could be in units of Euros without requiring an a priori trade-off between the two.

Weaknesses

1. Since a standard formulation of DEA creates a separate LP for each DMU, large problems can be computationally intensive.
2. Since DEA is an extreme point technique, noise (even symmetrical noise with zero mean) such as measurement error can cause significant problems.
3. DEA is good at estimating relative efficiency of a DMU but it converges very slowly to "absolute" efficiency. In other words, it can tell you how well you are doing compared to your peers but not compared to a "theoretical maximum".
4. Since DEA is a nonparametric technique, statistical hypothesis tests are difficult to apply in the DEA context.
5. DEA is very generous: If a DMU excels in just one output (or input) it is likely to get 100% efficiency, even if it performs very badly in all the other outputs (or inputs). So, if there are many outputs (or inputs) one is likely to get 100% efficiency for all the DMUs, which means that DEA cannot differentiate between the DMUs (it does not mean that all the DMUs are doing well in any absolute sense).

Chapter 10

Transportation Problems

In this chapter we shall briefly consider a set of so-called transportation problems that can be modelled as LPs, and thus solved with, say, the Simplex/Big M algorithm. There is, however, a specific structure in the models that allows us to use specialized algorithms that are much faster — and more pen-and-paper-friendly — than the Simplex/Big M algorithm.

This chapter is adapted from [2, Ch. 3], [4, Ch. 6], and T. S. Ferguson's web notes.

10.1 Transportation Algorithm

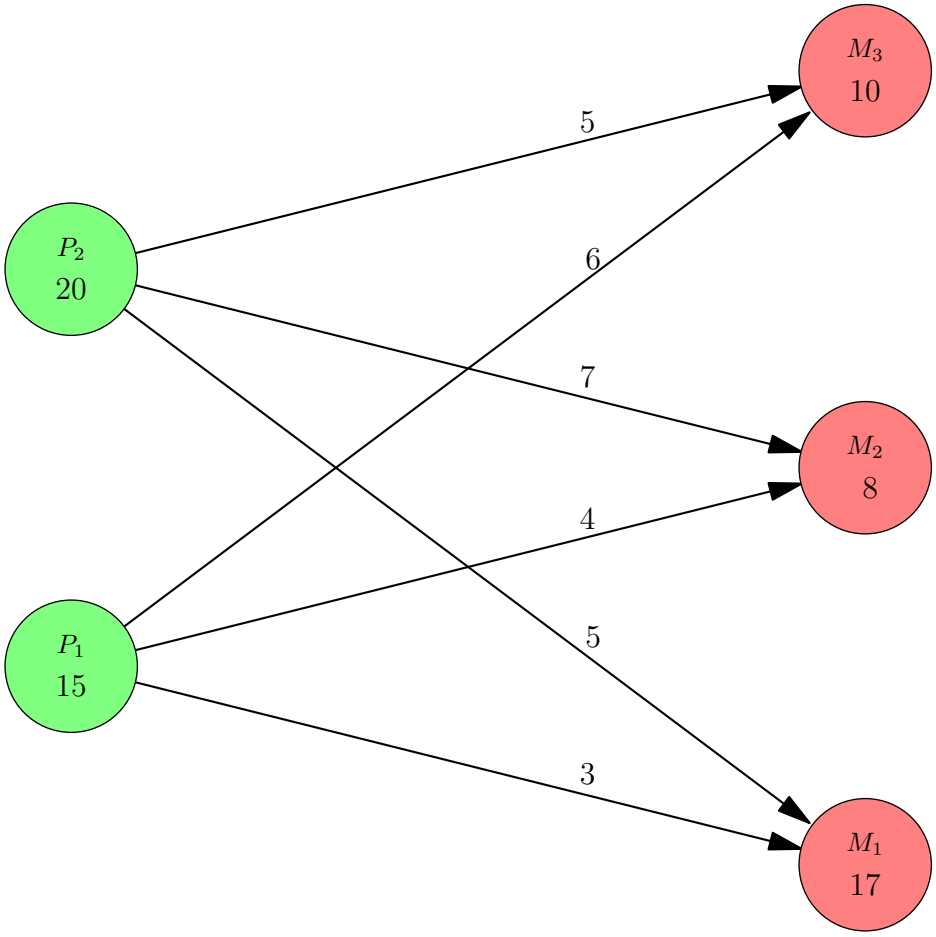
Transportation Problems as Linear Programs

10.1.1 Example. Frábært ehf. produces skyr. It has two production plants: P_1 and P_2. Plant P_1 produces 15 tons of skyr per day, and plant P_2 produces 20 tons of skyr per day. All the produced skyr is shipped to the markets M_1, M_2, and M_3. The market M_1 demands 17 tons of skyr, the market M_2 demands 8 tons of skyr, and the market M_3 demands 10 tons of skyr.

To ship one ton of skyr from plant P_1 to markets M_1, M_2, and M_3 costs €3, €4, and €6, respectively. To ship one ton of skyr from plant P_2 to markets M_1, M_2, and M_3 costs €5, €7, and €5, respectively.

Frábært ehf. wants to deliver its skyr to the markets with the minimal possible shipping cost while still meeting the market demands. How should Frábært ship its skyr?

The next picture illustrates the Frábært ehf.'s situation:



The Frábært's problem is a typical transportation problem, and it can be modelled as an LP.

Let

$$x_{ij} \quad = \quad \text{tons of skyr shipped from plant } i \text{ to market } j.$$

These are obviously the decision variables for Frábært. Everything else is fixed, and the only thing that is left open is the actual amounts transported from ports to markets.

The objective of any transportation problem is to minimize the total shipping cost (while meeting the market demand). So, Frábært's objective is

$$\min z \quad = \quad \sum_{i=1}^{2} \sum_{j=1}^{3} c_{ij} x_{ij} \tag{10.1.2}$$

where

$$c_{ij} \quad = \quad \text{the cost of shipping one ton of skyr from plant } i \text{ to market } j.$$

The objective (10.1.2) is a linear objective: Sums are linear, and double-sums doubly so.

What about the constraints then?

There are of course the *sign constraints*

$$x_{ij} \geq 0 \qquad \text{for all plants } i \text{ and markets } j.$$

Indeed, it would be pretty hard to transport negative amount of skyr.

There are also the supply and demand constraints.

Each plant P_i has only so many tons of skyr it can supply. So, if s_i is the supply limit for plant P_i then we have the *supply constraints*

$$\sum_{j=1}^{3} x_{ij} \leq s_i \qquad \text{for all plants } i.$$

Each market demands so many tons of skyr, and according to the problem we are committed to meet the market demands. So, if d_j is the demand for skyr in the market M_j then we have the *demand constraints*

$$\sum_{i=1}^{2} x_{ij} \geq d_j \qquad \text{for all markets } j.$$

Here is the LP for Frábært ehf.:

(10.1.3)

$$\begin{array}{lrrrrrrrr}
\min z = & 3x_{11} & +4x_{12} & +6x_{13} & +5x_{21} & +7x_{22} & +5x_{23} & & \\
\text{s.t.} & x_{11} & +x_{12} & +x_{13} & & & & \leq & 15 \\
 & & & & x_{21} & +x_{22} & +x_{23} & \leq & 20 \\
 & x_{11} & & & +x_{21} & & & \geq & 17 \\
 & & x_{12} & & & +x_{22} & & \geq & 8 \\
 & & & x_{13} & & & +x_{23} & \geq & 10 \\
 & & & & & & x_{ij} & \geq & 0
\end{array}$$

The LP (10.1.3) can be solved by using, e.g., the Simplex/Big M method. There is, however, a much more efficient specialized algorithm for solving LPs of type (10.1.3). We shall learn this method later in this Chapter. For now, let us end this subsection by noting the form of a general transportation problem as an LP and note its dual LP:

10.1.4 Definition. The *transportation problem* with

ports P_i, $i = 1, \ldots, m$,
markets M_j, $j = 1, \ldots, n$,
shipping costs c_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$,
supplies s_j, $j = 1, \ldots, m$, and

demands d_i, $i = 1, \ldots, n$

is

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad \text{(total shipping cost)}$$

subject to

$$\begin{aligned}
\sum_{j=1}^{n} x_{ij} &\leq s_i \qquad \text{for all } i = 1, \ldots, m, && \text{(supply)}\\
\sum_{i=1}^{m} x_{ij} &\geq d_j \qquad \text{for all } j = 1, \ldots, n, && \text{(demand)}\\
x_{ij} &\geq 0 \qquad \text{for all } i = 1, \ldots, m,\ j = 1, \ldots, n.
\end{aligned}$$

The dual LP of the transportation problem of Definition 10.1.4 is

$$\begin{aligned}
\max w = \ & \sum_{j=1}^{n} d_j v_j + \sum_{i=1}^{m} s_i u_i \\
\text{s.t.} \quad & v_j - u_i \leq c_{ij} \quad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n \\
& u_i, v_j \geq 0 \quad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n
\end{aligned} \tag{10.1.5}$$

Balancing Transportation Problems

Example 10.1.1 is *balanced*: Total supply equals total demand, i.e.,

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

This is essential in transportation problems. Also, note that with balanced problems it follows that all the products will be shipped, and that the market demands will be met exactly. So, the supply and demand constraints will realize as equalities rather than inequalities. So, we can, for balanced problems, restate Definition 10.1.4 as

10.1.6 Definition. The *transportation problem* with

ports P_i, $i = 1, \ldots, m$,
markets M_j, $j = 1, \ldots, n$,
shipping costs c_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$,
supplies s_j, $j = 1, \ldots, m$, and
demands d_i, $i = 1, \ldots, n$

is

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad \text{(total shipping cost)}$$

subject to

$$\begin{aligned}
\sum_{j=1}^{n} x_{ij} &= s_i \qquad \text{for all } i = 1, \ldots, m, && \text{(supply)} \\
\sum_{i=1}^{m} x_{ij} &= d_j \qquad \text{for all } j = 1, \ldots, n, && \text{(demand)} \\
x_{ij} &\geq 0 \qquad \text{for all } i = 1, \ldots, m,\ j = 1, \ldots, n.
\end{aligned}$$

What about non-balanced transportation problems then? There are two possible cases:

More supply than demand In this case we can introduce an imaginary market called the *dump*. The dump has just enough demand so that you can — dump — you excess supply there. Shipping to dump is costless. This solution may seem silly to you, but remember that the objective was to minimize the *transportation cost while meeting the market demand*. So, you are allowed to lose your shipments, if it helps.

More demand than supply In this case the problem is infeasible: You cannot meet the market demands if you do not have enough to supply. One can of course generalize the transportation problem so that if the supply does not meet the demand there is a (linear) penalty associated with each market whose demand is not satisfied. E.g., one could subtract value

$$M\left(d_j - \sum_{i=1}^{m} x_{ij}\right)$$

from the objective for each market j whose demand d_j is not met. We shall not consider penalized transportation problems here, however.

Specialized Algorithm for Transportation Problems

The transportation algorithm is a tableau-dance with transportation tableaux. An empty transportation tableau is a tabular representation of the transport problem's 10.1.4 data:

	M_1	M_2	$\cdots$	M_n	
P_1	c_{11}	c_{12}		c_{1n}	s_1
P_2	c_{21}	c_{22}		c_{2n}	s_2
$\vdots$					$\vdots$
P_m	c_{m1}	c_{m2}		c_{mn}	s_m
	d_1	d_2	$\cdots$	d_n	

The Frábært's transportation problem's 10.1.1 tabular representation is

	M_1	M_2	M_3	
P_1	3	4	6	15
P_2	5	7	5	20
	17	8	10	

The transportation algorithm fills the empty transportation tableau with the shipping schedule x_{ij}:

	M_1	M_2	$\cdots$	M_n	
P_1	c_{11} x_{11}	c_{12} x_{12}		c_{1n} x_{1n}	s_1
P_2	c_{21} x_{21}	c_{22} x_{22}		c_{2n} x_{2n}	s_2
$\vdots$					$\vdots$
P_m	c_{m1} x_{m1}	c_{m2} x_{m2}		c_{mn} x_{mn}	s_m
	d_1	d_2	$\cdots$	d_n	

The general idea of the specialized *Transportation Algorithm* is the same as in the Simplex algorithm, viz.

Meta-Step 1: Find a BFS.

Meta-Step 2: Check for optimality. If solution is optimal, the algorithm terminates. Otherwise move to step 3.

Meta-Step 3: Find a new, improved, BFS. Go back to Meta-Step 2.

Next we explain the metas away from the Meta-Steps 1–3 above.

Finding a BFS The first BFS can be found, e.g., by using the so-called *NW Corner Method*. The method is called such since the traditional way of choosing available squares goes from NW to SE. The method goes as follows:

1. Choose any available square, say (i_0, j_0). Specify the shipment x_{i_0,j_0} as large as possible subject to the supply and demand constraints, and mark this variable.
2. Delete from consideration whichever row or column has its constraint satisfied, but not both. If there is a choice, do not delete a row (column) if it is the last row (resp. column) undeleted.
3. Repeat 1. and 2. until the last available square is filled with a marked variable, and then delete from consideration both row and column.

Now we construct the first BFS for Example 10.1.1. The marked variables will be in parentheses, and the deleted squares will have 0.

We start with the NW corner. So $(i_0, j_0) = (1, 1)$. We put there as a big number as possible. This means that we ship all the plant P_1's supply to market M_1, i.e., $x_{11} = 15$. Now there is nothing left in P_1. So, we must have $x_{12} = x_{13} = 0$. So, squares $(1, 2)$ and $(1, 3)$ get deleted — or, if you like — row 1 gets deleted.

	M_1	M_2	M_3	
P_1	3 (15)	4 0	6 0	15
P_2	5	7	5	20
	17	8	10	

Next we move south, since east is deleted. The biggest number we can now put to square $(2, 1)$ is $x_{21} = 2$, since there is already 15 tons of skyr shipped to the market M_1 that demands 17 tons. No rows or columns will get deleted because of this operation.

	M_1	M_2	M_3	
P_1	3 (15)	4 0	6 0	15
P_2	5 (2)	7	5	20
	17	8	10	

Next we move east to square $(2,2)$. There the biggest number we can put is $x_{22} = 8$ since that is the market demand for M_2. No rows or columns will be deleted because of this operation.

	M_1	M_2	M_3	
P_1	3 (15)	4 0	6 0	15
P_2	5 (2)	7 (8)	5	20
	17	8	10	

Finally, we move to the last free square $(2,3)$. It is obvious that $x_{23} = 10$. We get the first BFS:

	M_1	M_2	M_3	
P_1	3 (15)	4 0	6 0	15
P_2	5 (2)	7 (8)	5 (10)	20
	17	8	10	

Checking Optimality Given a BFS, i.e., a feasible shipping schedule x_{ij}, we shall use the Complementary Slackness Theorem 8.2.20 to check whether the BFS is optimal. This means finding dual variables u_i and v_j that satisfy

$$x_{ij} > 0 \quad \text{implies that} \quad v_j - u_i = c_{ij}.$$

One method of finding the dual variables u_i and v_j is to solve the equations

$$v_j - u_i \;=\; c_{ij}$$

for all (i,j)-squares containing marked variables. There are $m+n-1$ marked variables, and so we have $m+n-1$ equations with $m+n$ unknowns. This means that one of the variables u_i, v_j can be fixed to 0, say. Some of the u_i or v_j may turn out to be negative, which as such is not allowed in the dual problem, but this is not a problem. Indeed, one can always add a big enough constant to all the u_is and v_js without changing the values of $v_j - u_i$.

Once the dual variables u_i and v_j are found we can check the optimality of the BFS by using the following algorithm:

1. Set one of the v_j or u_i to zero, and use the condition

$$v_j - u_i \quad = \quad c_{ij}$$

for *squares containing marked variables* to find all the v_j and u_i.

2. Check feasibility,

$$v_j - u_i \quad \leq \quad c_{ij},$$

for the *remaining squares*. If the BFS is feasible, it is optimal for the problem and its dual, due to the Complementary Slackness Theorem 8.2.20

Let us then find the dual variables u_i and v_j for the BFS Frábært's problem 10.1.1. The next tableau is the BFS we found with the, yet unknown, dual variables in their appropriate places.

	v_1	v_2	v_3	
u_1	3 (15)	4 0	6 0	15
u_2	5 (2)	7 (8)	5 (10)	20
	17	8	10	

To solve u_i and v_j for the marked variables we put $u_2 = 0$. Remember we can choose any one of the u_i or v_j be zero. We chose u_2 because then we see immediately that $v_1 = 5$, $v_2 = 7$, and $v_3 = 5$. As for the last unknown u_1, we have

$$u_1 \quad = \quad v_1 - c_{11} \quad = \quad 5 - 3 \quad = \quad 2.$$

So, we have the following BFS with the duals

	5	7	5	
2	3 (15)	4 0	6 0	15
0	5 (2)	7 (8)	5 (10)	20
	17	8	10	

Now we check the remaining squares. For the BFS to be optimal we must have $v_j - u_i \leq c_{ij}$. We see that this is not the case. The culprit is the square $(1, 2)$:

$$v_2 - u_1 \quad = \quad 7 - 2 \quad = \quad 5 \quad > \quad 4 \quad = \quad c_{12}.$$

This means that we must improve our BFS.

Improvement Routine Now we have a BFS that is not optimal. So, we must have a square (i_0, j_0), say, for which

$$v_{j_0} - u_{i_0} \quad > \quad c_{i_0 j_0}.$$

We would like to ship some amount of skyr, say, from the port P_{i_0} to the market M_{j_0}. The current amount $x_{i_0 j_0} = 0$ will be changed to a new amount denoted by Δ. But if we change $x_{i_0 j_0}$ to Δ we must subtract and add Δ to other squares containing marked variables. This means that we are looking forward to a new BFS

	M_1	M_2	M_3	
P_1	3 $-\Delta$ (15)	4 $+\Delta$ 0	6 0	15
P_2	5 $+\Delta$ (2)	7 $-\Delta$ (8)	5 (10)	20
	17	8	10	

Now we choose the change Δ to be as big as possible bearing in mind that the shipments cannot be negative. This means that Δ will be the minimum of the x_{ij}s in the squares we are subtracting Δ. We see that the biggest possible change is $\Delta = 8$, which makes the the shipment x_{22} zero.

10.1.7 Remark. It may turn out that that we have $\Delta = 0$. This means that the value of the objective won't change. However, the shipping schedule and the marked variables will change. While the new shipping schedule is no better than the old one, one can hope that from this new shipping schedule one can improve to a better one.

Now we have the new BFS

	M_1	M_2	M_3	
P_1	3 (7)	4 (8)	6 0	15
P_2	5 (10)	7 0	5 (10)	20
	17	8	10	

We have to go back to the previous step and check optimality for this BFS. So, we have to solve the dual variables u_i and v_j. We set now $u_2 = 0$ which gives us immediately that $v_1 = 5$ and $v_3 = 5$. So, we find out that

$$u_1 \;=\; v_1 - c_{11} \;=\; 2,$$

and that

$$v_2 \;=\; u_1 + c_{12} \;=\; 6.$$

So, we have the BFS with dual variables

	5	6	5	
2	3 (7)	4 (8)	6 0	15
0	5 (10)	7 0	5 (10)	20
	17	8	10	

This solutions passes the optimality test:

$$v_j - u_i \quad \leq \quad c_{ij}$$

for all i and j. So, we have found an optimal shipping schedule. The cost associated with this shipping schedule can now be easily read from the tableau above:

$$\begin{aligned} z &= \sum_{i=1}^{2}\sum_{j=1}^{3} c_{ij}x_{ij}^{*} \\ &= 3\times 7 + 4\times 8 + 5\times 10 + 5\times 10 \\ &= 153. \end{aligned}$$

The improvement algorithm can now be stated as

1. Choose any square (i, j) with $v_j - u_i > c_{ij}$. Set $x_{ij} = \Delta$, but keep the constraints satisfied by subtracting and adding Δ to appropriate marked variables.
2. Choose Δ to be the minimum of the variables in the squares in which Δ is subtracted.
3. Mark the new variable x_{ij} and remove from the marked variables one of the variables from which Δ was subtracted that is now zero.

10.2 Assignment Problem

In this section we consider assignment problems that are — although it may not seem so at first sight — special cases of transportation problems.

Assignment Problems as Linear Programs

10.2.1 Example. Machineco has four machines and four jobs to be completed. Each machine must be assigned to complete one job. The times required to set up each machine for completing each job are:

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

Machineco wants to minimize the total setup time needed to complete the four jobs.

How can LP be used to solve Machineco's problem?

The key point in modelling the Machineco's problem 10.2.1 is to find out the decision variables — everything else is easy after that. So what are the decisions Machineco must make? Machineco must choose which machine is assigned to which job. Now, how could we write this analytically with variables? A common trick here is to use *binary variables*, i.e., variables that can take only two possible values: 0 or 1. So, we set binary variables x_{ij}, $i = 1, \ldots, 4$, $j = 1, \ldots, 4$, for each machine and each job to be

$$\begin{aligned} x_{ij} &= 1 \text{ if machine } i \text{ is assigned to meet the demands of job } j, \\ x_{ij} &= 0 \text{ if machine } i \text{ is } \textit{not} \text{ assigned to meet the demands of job } j. \end{aligned}$$

In other words, the variable x_{ij} is an *indicator* of the claim

"Machine i is assigned to job j".

Now it is fairly easy to formulate a program, i.e., an optimization problem, for Machineco's problem 10.2.1. Indeed, the objective is to minimize the total

setup time. With our binary variables we can write the total setup time as

$$\begin{aligned} z \;=\; & 14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} \\ & +2x_{21} + 12x_{22} + 6x_{23} + 4x_{24} \\ & +7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} \\ & +2x_{41} + 4x_{42} + 6x_{43} + 10x_{44} \end{aligned}$$

Note that there will be a lot of zeros in the objective function above.

What about the constraints for Machineco? First, we have to ensure that each machine is assigned to a job. This will give us the supply constraints

$$\begin{aligned} x_{11} + x_{12} + x_{13} + x_{14} &= 1 \\ x_{21} + x_{22} + x_{23} + x_{24} &= 1 \\ x_{31} + x_{32} + x_{33} + x_{34} &= 1 \\ x_{41} + x_{42} + x_{43} + x_{44} &= 1 \end{aligned}$$

Second, we have to ensure that each job is completed, i.e., each job has a machine assigned to it. This will give us the demand constraints

$$\begin{aligned} x_{11} + x_{21} + x_{31} + x_{41} &= 1 \\ x_{12} + x_{22} + x_{32} + x_{42} &= 1 \\ x_{13} + x_{23} + x_{33} + x_{43} &= 1 \\ x_{14} + x_{24} + x_{34} + x_{44} &= 1 \end{aligned}$$

So, putting the objective and the constraints we have just found together, and not forgetting the binary nature of the decisions, we have obtained the following program for Machineco's problem 10.2.1:

$$\begin{aligned} \min z = \; & 14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} \\ & +2x_{21} + 12x_{22} + 6x_{23} + 4x_{24} \\ & +7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} \\ & +2x_{41} + 4x_{42} + 6x_{43} + 10x_{44} \\ \text{s.t.} \quad & x_{11} + x_{12} + x_{13} + x_{14} = 1 \quad \text{(Machine)} \\ & x_{21} + x_{22} + x_{23} + x_{24} = 1 \\ & x_{31} + x_{32} + x_{33} + x_{34} = 1 \\ & x_{41} + x_{42} + x_{43} + x_{44} = 1 \\ & x_{11} + x_{21} + x_{31} + x_{41} = 1 \quad \text{(Job)} \\ & x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ & x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ & x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ & x_{14} + x_{24} + x_{34} + x_{44} = 1 \\ & x_{ij} = 0 \quad \text{or} \quad x_{ij} = 1 \end{aligned} \tag{10.2.2}$$

In (10.2.2) we have *binary constraints* $x_{ij} = 0$ or $x_{ij} = 1$ for the decision variables. So, at first sight it seems that the program (10.2.2) is not a linear

one. However, the structure of the assignment problems is such that if one omits the assumption $x_{ij} = 0$ or $x_{ij} = 1$, and simply assumes that $x_{ij} \geq 0$, one will get on optimal solution where the decisions x_{ij}^* are either 0 or 1. Hence, the program (10.2.2) is a linear one, i.e., it is an LP. Or, to be more precise, the program (10.2.2) and its linear relaxation

$$
\begin{array}{lrcl}
\min z = & 14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} & & \\
& +2x_{21} + 12x_{22} + 6x_{23} + 4x_{24} & & \\
& +7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} & & \\
& +2x_{41} + 4x_{42} + 6x_{43} + 10x_{44} & & \\
\text{s.t.} & x_{11} + x_{12} + x_{13} + x_{14} & = 1 & \text{(Machine)} \\
& x_{21} + x_{22} + x_{23} + x_{24} & = 1 & \\
& x_{31} + x_{32} + x_{33} + x_{34} & = 1 & \\
& x_{41} + x_{42} + x_{43} + x_{44} & = 1 & \\
& x_{11} + x_{21} + x_{31} + x_{41} & = 1 & \text{(Job)} \\
& x_{11} + x_{21} + x_{31} + x_{41} & = 1 & \\
& x_{12} + x_{22} + x_{32} + x_{42} & = 1 & \\
& x_{13} + x_{23} + x_{33} + x_{43} & = 1 & \\
& x_{14} + x_{24} + x_{34} + x_{44} & = 1 & \\
& x_{ij} & \geq 0 &
\end{array}
\tag{10.2.3}
$$

are equivalent. Equivalence of programs means that they have the same optimal decision and the same optimal objective value.

Now, notice that the assignment LP (10.2.3) can be considered as a transportation problem. Indeed, consider the four machines as ports and the four jobs as markets. Then the supplies and demands are all ones. This interpretation gives us the following definition of an assignment problem:

10.2.4 Definition. An *assignment problem* is a transportation problem with equal amount of ports and markets, where the demands and supplies for each port and market are equal to one.

Since assignment problems are LPs they can be solved as any LP with the Simplex/Big M method, say. Also, since they are transportation problems, they can be solved by using the specialized transportation algorithm. There is, however, an even more specialized algorithm for transportation problems: The so-called Hungarian method.

Hungarian Method

In the Hungarian method, the data of the assignment problem is presented in an $n \times n$-table, where n is the number of ports, or markets, which is the same. For Example 10.2.1 the "Hungarian tableau" is

14	5	8	7
2	12	6	5
7	8	3	9
2	4	6	10

The Hungarian method is based on the following two observation:

1. The problem is solved if we can choose n squares from the "Hungarian tableau" so that:
   (a) Exactly one square is chosen from each row.
   (b) Exactly one square is chosen from each column.
   (c) The sum of the costs in the chosen n squares is the smallest possible.
2. If a same number is subtracted from all squares in a row, then the optimal selection of squares does not change. The same is true, if a same number is subtracted from all squares in a column.

Here is the *Hungarian Algorithm*:

Step 1 For each row, subtract the row minimum from each element in the row.

Step 2 For each column, subtract the column minimum from each element in the column.

Step 3 Draw the minimum number of lines — horizontal, vertical, or both — that are needed to cover all the zeros in the tableau. If n lines were required then the optimal assignment can be found among the covered zeros in the tableau, and the optimal cost can be read from the first tableau by summing up the number in the squares corresponding to the lined-out zeros.

Step 4 Find the smallest non-zero element that is uncovered by lines. Subtract this element from each uncovered element, and add this element to each square that is covered with two lines. Return to Step 3.

Here are steps 1 and 2 for Example 10.2.1:

14	5	8	7
2	12	6	5
7	8	3	9
2	4	6	10

$\rightsquigarrow$

9	0	3	2
0	10	4	3
4	5	0	6
0	2	4	8

$\rightsquigarrow$

9	0	3	0
0	10	4	1
4	5	0	4
0	2	4	6

Here are the remaining steps 3 and 4 for Example 10.2.1:

9	0	3	0
0	10	4	1
4	5	0	4
0	2	4	6

$\rightsquigarrow$

10	0	4	0
0	9	4	0
4	4	0	3
0	1	4	5

Now, we go back to Step 3, and line-out zeros:

10	0	4	0
0	9	4	0
4	4	0	3
0	1	4	5

Now we can read the optimal assignment from the covered zeros. First we note that the only covered zero in column 3 is in square $(3,3)$. So, we must have assignment $x_{33} = 1$. Also, in column 2 the only available zero is in square $(1,2)$. Consequently, $x_{12} = 1$. Now, as we can no longer use row 1 the only available zero in column 4 is in square $(2,4)$. So, $x_{24} = 1$. Finally, we choose $x_{41} = 1$. Next we read the corresponding setup cost from the first Hungarian tableau. We obtain

$$\text{setup cost} \quad = \quad 5 + 5 + 3 + 2 \quad = \quad 15.$$

10.3 Transshipment Problem

In this section we consider transportation problems. At first sight the transportation problems are network problems. However, one can think of them as generalizations of transportation problems. In that sense we are considering here problems that are opposite to assignment problems in the sense that assignment problems are special cases of transportation problems and transportation problems are special cases of transshipment problems. It turns out, however, that we can express transshipment problems as transportation problems. So the generalization from transportation problems to transshipment problems is mathematically no generalization at all.

Transshipment Problems as Transportation Problems

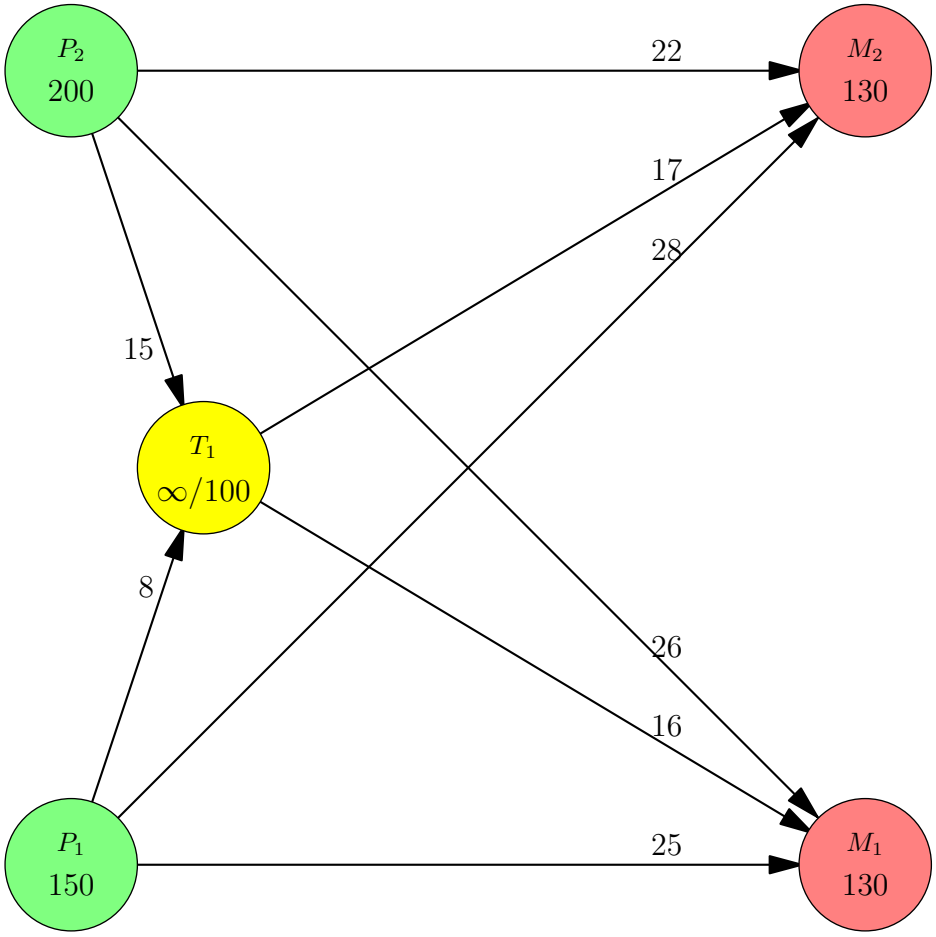
10.3.1 Example. The Generic Company produces generic products and ships them to the markets M_1 and M_2 from the ports P_1 and P_2. The products can be either shipped directly to the markets, or the Generic Company can use a transshipment point T_1.

The ports P_1 and P_2 supply 150 and 200 units, respectively. The markets M_1 and M_2 demand both 130 units. The shipment costs from P_1 to M_1 is 25, from P_1 to M_2 it is 28, from P_2 to M_1 it is 26, and from P_2 to M_2 it is 22. The shipping cost from P_1 to the transshipment point T_1 is 8 and the shipping cost from P_2 to the transshipment point T_1 is 15. The shipping costs from the transshipment point T_1 to the markets M_1 and M_2 are 16 and 17, respectively.

The Generic Company wants to minimize its shipping costs while meeting the market demands. How should the Generic Company ship its products, if

(a) the transshipment point T_1 has infinite capacity,
(b) only 100 products can be transshipped through T_1?

Here is the data of Example 10.3.1 in a graph form.



The general idea in modelling transshipment problems is to model them as transportation problems where the transshipment points are both ports and markets. To be more precise:

- A transshipment problem is a transportation problem, where each transshipment point is both a port and a market. The shipping cost from a transshipment point to itself is, of course, zero.
- If a transshipment point has capacity N, then N will be both demand and supply for that transshipment point. If the capacity is unlimited set the demand and supply to be the total supply of the system for that transshipment point.
- Cost for shipping from transshipment points to the balancing dump is a very very very very big number M. This ensures that the excess supply to be dumped is dumped immediately.

Solving Transshipment Problems

Solution to Variant (a) of Example 10.3.1 In Example 10.3.1 the total supply is 350 and the total demand is 260. So, in order to balance the problem we need the dump market with demand 90. Let us denote the dump by D. The supply and demand for the transshipment point will be 350, so that everything can be, if necessary, transported via T_1. We also do not want to transship anything to the dump via the transshipment point T_1. So, we make the cost of shipping from T_1 to D a very very very very very big number M. The direct dumping from P_1 or P_2 to D of costless, of course. So, we have the following initial transportation tableau

	T_1	M_1	M_2	D	
P_1	8	25	28	0	150
P_2	15	26	22	0	200
T_1	0	16	17	M	350
	350	130	130	90	

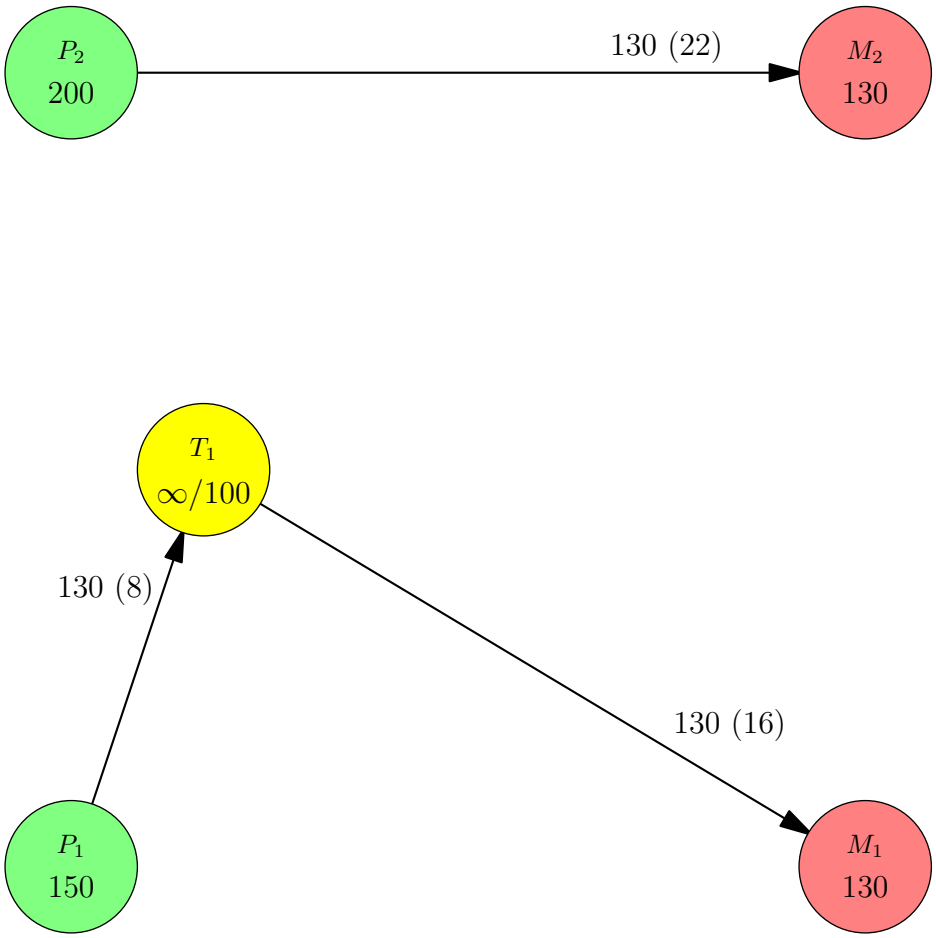
Now we can apply the transportation algorithm to this tableau. Here is a (greedy minimal cost) initial BFS

	T_1	M_1	M_2	D	
P_1	8 (150)	25 0	28 0	0 0	150
P_2	15 (110)	26 0	22 0	0 (90)	200
T_1	0 (90)	16 (130)	17 (130)	M 0	350
	350	130	130	90	

We leave it as an exercise to carry out the transportation algorithm for this BFS. We note the optimal transportation tableau:

	T_1	M_1	M_2	D	
P_1	8 (130)	25 0	28 0	0 (20)	150
P_2	15 0	26 0	22 (130)	0 (70)	200
T_1	0 (220)	16 (130)	17 0	M 0	350
	350	130	130	90	

Here is a graphical representation of the optimal transportation tableau above (the costs are in parentheses):



From the picture we see that the total shipping cost is

$$130 \times 8 + 130 \times 16 + 130 \times 22 \quad = \quad 5980.$$

We cannot read the transshipment and dump data of the last Transportation Tableau from the picture above, but that data was auxiliary anyway, i.e. we might well not be interested in it.

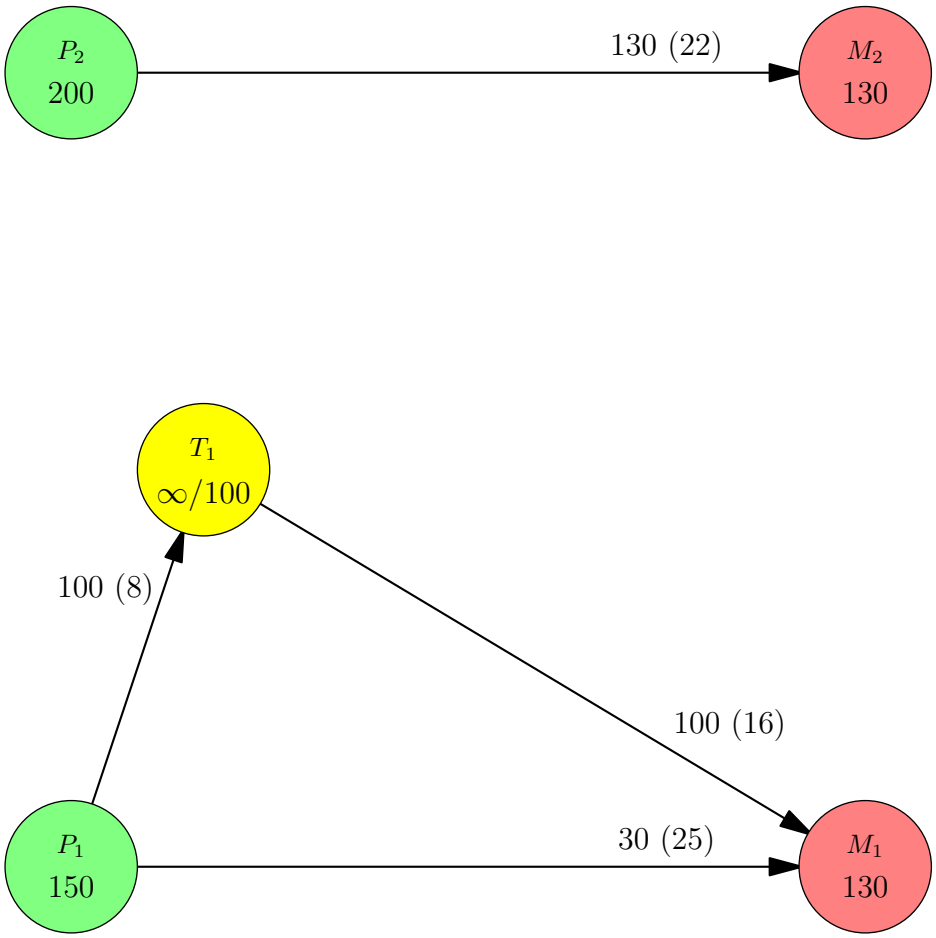
Solution to Variant (b) of Example 10.3.1 In Example 10.3.1 the total supply is 350 and the total demand is 260. So, in order to balance the problem we need the dump market with demand 90. Let us denote the dump by D. The supply and demand for the transshipment point will be 100, so that at most 100 units can be transported via T_1. We also do not want to transship anything to the dump via the transshipment point T_1. So, we make the cost of shipping from T_1 to D a very very very very very big number M. The direct dumping from P_1 or P_2 to D of costless, of course. So, we have the following initial transportation tableau

	T_1	M_1	M_2	D	
P_1	8	25	28	0	150
P_2	15	26	22	0	200
T_1	0	16	17	M	100
	100	130	130	90	

We leave it for the students to carry out the transportation algorithm, and simply note the solution:

	T_1	M_1	M_2	D	
P_1	8 (100)	25 (30)	28 0	0 (20)	150
P_2	15 0	26 0	22 (130)	0 (70)	200
T_1	0 0	16 (100)	17 0	M 0	100
	100	130	130	90	

Here is a graphical representation of the optimal transportation tableau above (the costs are in parentheses):



From the picture we see that the total shipping cost is

$$100 \times 8 + 100 \times 16 + 30 \times 25 + 130 \times 22 \quad = \quad 6010.$$

We cannot read the transshipment and dump data of the last Transportation Tableau from the picture above, but that data was auxiliary anyway, i.e. we might well not be interested in it.

Part IV

Non-Linear Programming

Chapter 11

Integer Programming

In this chapter we lift the *divisibility* assumption of LPs. This means that we are looking at LPs where some (or all) of the decision variables are required to be integers. No longer can Tela Inc. from Example 2.1.1 in Chapter 2 manufacture 566.667 number of product #1.

This chapter is adapted from [4, Ch. 8].

11.1 Integer Programming Terminology

Integer Programs' Linear Relaxations

Although the name *Integer Program* (IP) does not state it explicitly, it is assumed that

> *IP is an LP with the additional requirement that some of the decision variables are integers.*

If the additional requirement that some of the decision variables are integers is lifted, then the resulting LP is called the *LP relaxation* of the IP in question.

Pure Integer Programs

An IP in which *all* the decision variables are required to be integers is called a *Pure Integer Program*, or simply an *Integer Program* (IP). For example,

$$\begin{array}{lrcrcrcl} & \max z = & 3x_1 & + & 2x_2 & & & \\ \text{(11.1.1)} & \text{s.t.} & x_1 & + & x_2 & \leq & 6 & \\ & & & & x_1, x_2 & \geq & 0, & \text{integer} \end{array}$$

is a pure integer program.

Mixed Integer Programs

An IP in which *some but not all* of the decision variables are required to be integers is called a *Mixed Integer Program* (MIP). For example,

$$\begin{array}{lrcrcrl} & \max z = & 3x_1 & + & 2x_2 & & \\ (11.1.2) & \text{s.t.} & x_1 & + & x_2 & \leq & 6 \\ & & & & x_1, x_2 & \geq & 0, x_1 \text{ integer} \end{array}$$

is a pure integer program. Indeed, x_1 is required to be an integer, but x_2 is not.

11.2 Branch-And-Bound Method

In this section we provide a relatively fast algorithm for solving IPs and MIPs. The general idea of the algorithm is to solve LP relaxations of the IP and to look for an integer solution by branching and bounding on the decision variables provided by the the LP relaxations.

Branch-And-Bound by Example

Let us solve the following pure IP:

11.2.1 Example. The Integrity Furniture Corporation manufactures tables and chairs. A table requires 1 hour of labor and 9 units of wood. A chair requires 1 hour of labor and 5 units of wood. Currently 6 hours of labor and 45 units of wood are available. Each table contributes €8 to profit, and each chair contributes €5 to profit.

Formulate and solve an IP to maximize Integrity Furniture's profit.

Let

$$\begin{array}{rcl} x_1 & = & \text{number of tables manufactured,} \\ x_2 & = & \text{number of chairs manufactured.} \end{array}$$

Since x_1 and x_2 must be integers, Integrity Furniture wishes to solve the following (pure) IP:

$$\begin{array}{lrcrcrll} & \max z = & 8x_1 & + & 5x_2 & & & \\ & \text{s.t.} & x_1 & + & x_2 & \leq & 6 & \text{(Labor)} \\ (11.2.2) & & 9x_1 & + & 5x_2 & \leq & 45 & \text{(Wood)} \\ & & & & x_1, x_2 & \geq & 0, \text{ integer} & \end{array}$$

The first step in the branch-and-bound method is to solve the LP relaxation of the IP (11.2.2):

$$
\begin{array}{lrcrcll}
\max z = & 8x_1 & + & 5x_2 & & & \\
\text{s.t.} & x_1 & + & x_2 & \leq & 6 & \text{(Labor)} \\
& 9x_1 & + & 5x_2 & \leq & 45 & \text{(Wood)} \\
& & & x_1, x_2 & \geq & 0 &
\end{array}
\tag{11.2.3}
$$

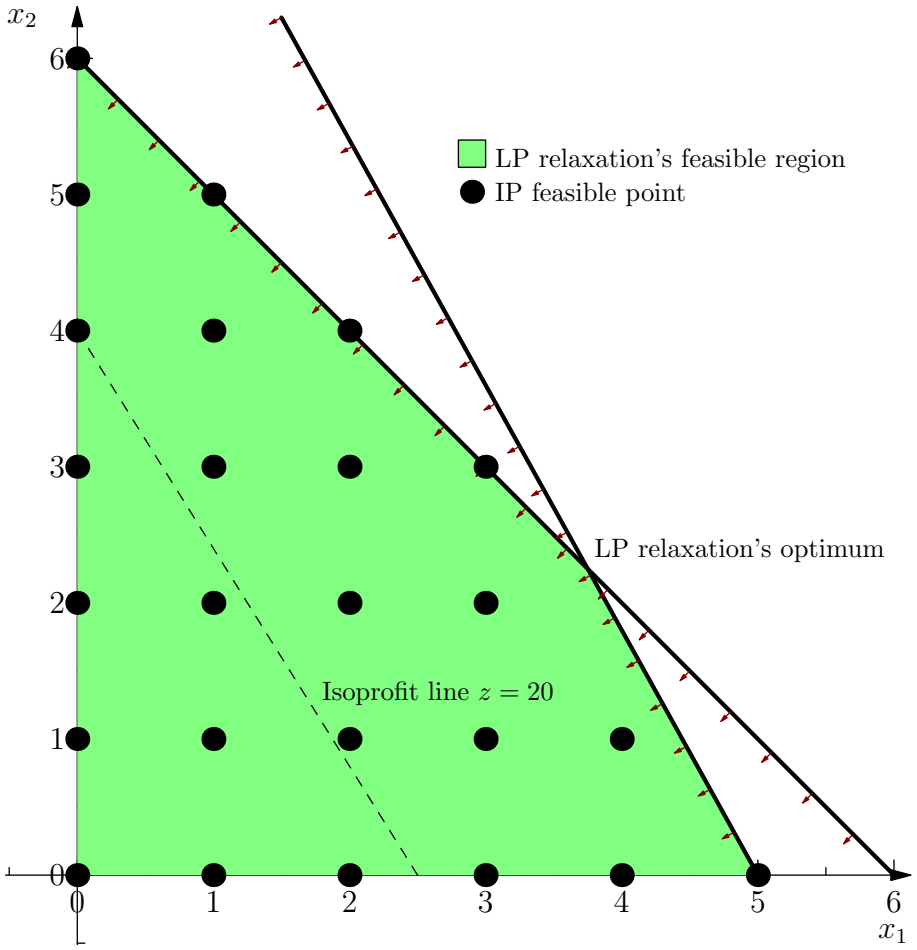
If all the decision variables (x_1 for tables and x_2 for chairs in the example we are considering) in the LP relaxation's optimum happen to be integers, then the optimal solution of the LP relaxation is also the optimal solution to the original IP.

In the branch-and-bound algorithm, we call the LP relaxation (11.2.3) of the IP (11.2.2) *subproblem 1* (SB 1).

After solving SP 1 (graphically, say, cf. the next picture) we find the solution to be

$$
\begin{aligned}
z &= 165/4 \\
x_1 &= 15/4 \\
x_2 &= 9/4
\end{aligned}
$$

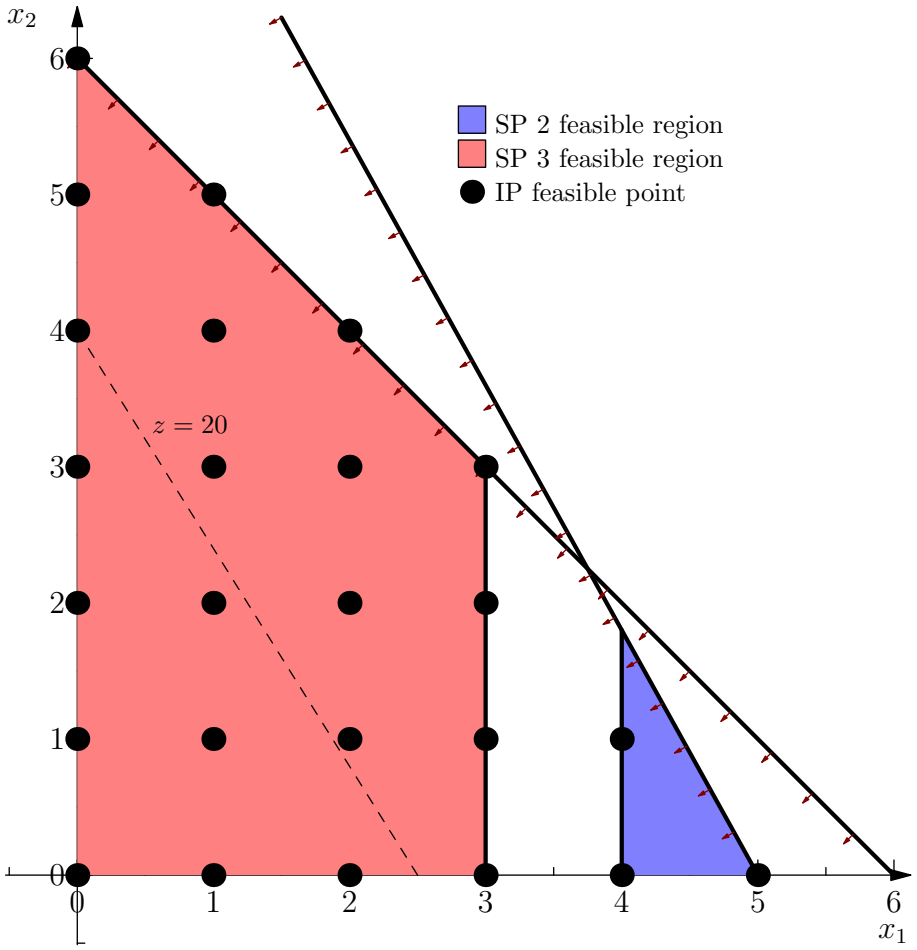
This means that we were not lucky: The decision variables turned out to be fractional. So, the LP relaxation (11.2.3) has (possibly) a better optimum than the original IP (11.2.2). In any case, we have found an *upper bound* to the original IP: Integrity Furniture Corporation cannot possibly have better profit than €165/4.



Next we split the feasible region (painted light green in the previous picture) of the LP relaxation (11.2.3) in hope to find a solution that is an integer one. We arbitrarily choose a variable that is fractional at the optimal solution of the LP SP 1 (the LP relaxation). We choose x_1. Now, x_1 was $15/4 = 3.75$. Obviously, at the optimal solution to the IP we have either $x_1 \leq 3$ or $x_1 \geq 4$, since the third alternative $3 < x_1 < 4$ is out of the question for IPs. So, we consider the two possible cases $x_1 \leq 3$ and $x_1 \geq 4$ as separate subproblems. We denote these subproblems as SP 2 and SP 3. So,

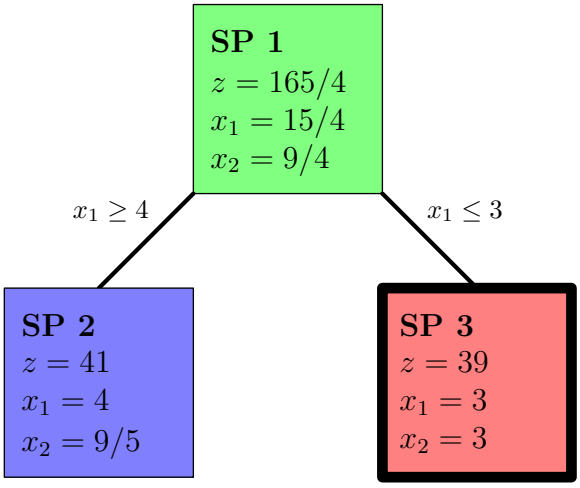
$$\begin{aligned} SP\ 2 &= SP\ 1 + "x_1 \geq 4", \\ SP\ 3 &= SP\ 1 + "x_1 \leq 3". \end{aligned}$$

In the next picture we see that every possible feasible solution of the Integrity Furniture's IP (the bullet points) is included in the feasible region of either SP 2 or SP 3. Also, SP 2 and SP 3 have no common points. Since SP 2 and SP 3 were created by adding constraints involving the fractional solution x_1, we say that SP 2 and SP 3 were created by *branching* on x_1.



We see that SP 3 has an integer solution. Unfortunately, the integer solution of SP 3, $z = 39, x_1 = 3, x_2 = 3$ is suboptimal when compared to the non-integer solution, $z = 41, x_! = 4, x_2 = 9/4$, of SP 2. So, it is possible that the SP 2 has a further subproblem that has better integer solution than SP 3. So, we have to branch SP 2.

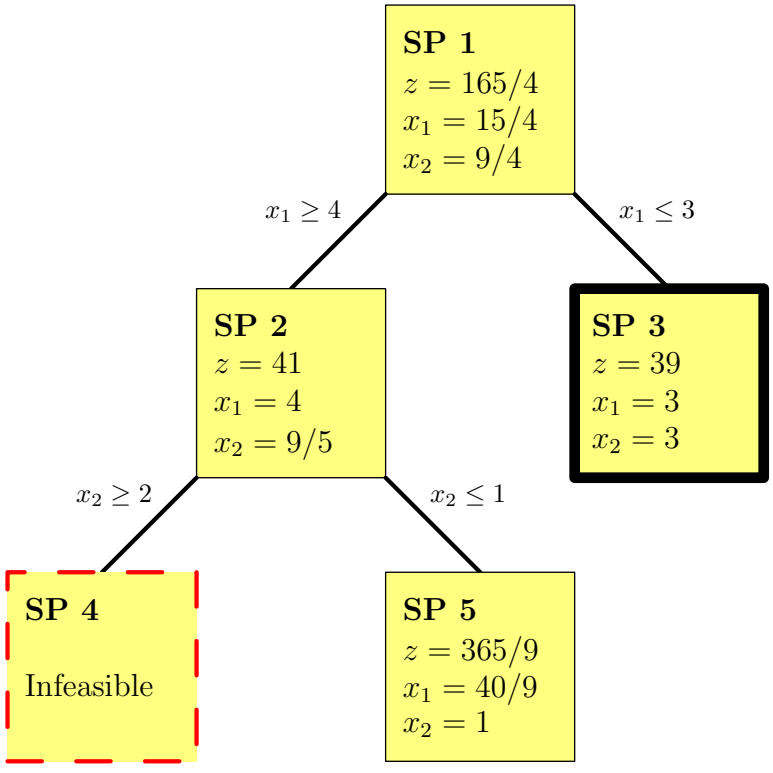
Before branching SP 2 let us represent what we have done in a tree. The colors in the next tree refer to the colors in the previous picture.



As pointed out, we must now branch on SP 2. Since $x_1 = 4$ is an integer. we branch on $x_2 = 9/5 = 1.8$. So, we have the new subproblems

$$\begin{aligned} SP\ 4 &= SP\ 2\ +\ "x_2 \geq 2", \\ SP\ 5 &= SP\ 2\ +\ "x_2 \leq 1". \end{aligned}$$

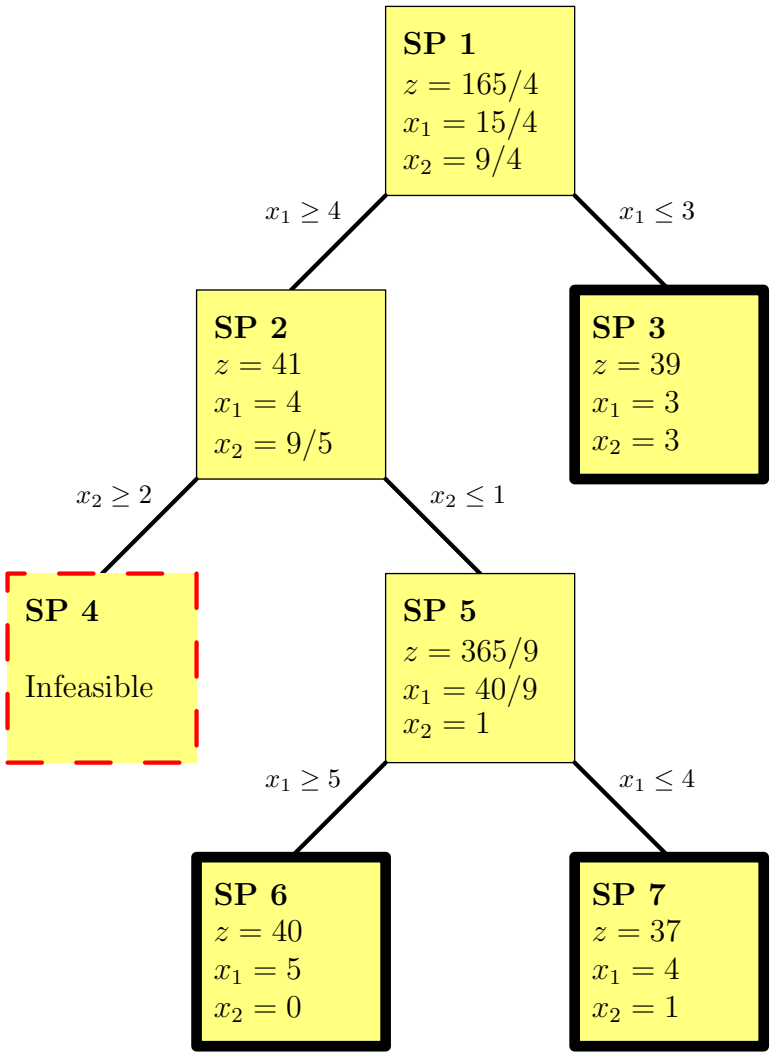
When the solutions to these subproblems are added to the tree above we get the following tree (the color coding is dropped):



We see that SP 4 is infeasible. So, the optimal solution is not there. However, SP 5 gives us a non-integer solution that is better than the integer solution of SP 3. So, we have to branch on SP 5. Since $x_2 = 1$ is already an integer, we branch on $x_1 = 40/9 = 4.444$. So, we get two new subproblems

$$\begin{aligned} SP\ 6 &= SP\ 5 + "x_1 \geq 5", \\ SP\ 7 &= SP\ 5 + "x_1 \leq 4". \end{aligned}$$

After this branching we finally arrive at the final solution where all the subproblems are either unfeasible. (There is no color coding in the boxes in the following tree. There is border coding, however. A thick borderline expresses a feasible IP solution, and a dashing red borderline expresses an infeasible case.)



From the tree above can read the solution to the IP: The SP 6 is the optimal

subproblem with integer solution. So, the solution to the IP (11.2.2) is

$$\begin{aligned} z &= 40, \\ x_1 &= 5, \\ x_2 &= 0. \end{aligned}$$

General Branch-And-Bound Algorithm

We have solved a pure IP with branch and bound. To solve MIP with branch-and-bound one follows the same steps as in the pure IP case *except* one only branches on decision variables that are required to be integers. So, solving MIPs is actually somewhat easier than solving pure IPs!

We have seen all the parts of the branch-and-bound algorithm. The essence of the algorithm is as follows:

1. Solve the LP relaxation of the problem. If the solution is integer where required, then we are done. Otherwise create two new subproblems by branching on a fractional variable that is required to be integer.

2. A subproblem is *not active* when any of the following occurs:

   (a) You used the subproblem to branch on.

   (b) All variables in the solution that are required to be integers, are integers.

   (c) The subproblem is infeasible.

   (d) You can fathom the subproblem by a bounding argument.

3. Choose an *active* subproblem and branch on a fractional variable that should be integer in the final solution. Repeat until there are no active subproblems.

4. Solution to the (M)IP is the best (M)IP solution of the subproblems you have created. It is found in one of the leafs of the tree representing the subproblems.

That's all there is to branch and bound!

11.3 Solving Integer Programs with GNU Linear Programming Kit

Solving (pure) IPs and MIPs with GLPK is very easy. The GLPK has all the necessary routines implemented and all you have to do is to declare which variables are required to be integers. To declare a decision variable as integer-valued one simply uses the keyword `integer` in the declaration.

Here is the GNU MathProg code for the simple IP (11.1.1):

```
#
# The IP (11.1.1)
#

# Decision variables
# Both x1 and x2 are required to be integers
var x1 >=0, integer;
var x2 >=0, integer;

# Objective
maximize z: 3*x1 + 2*x2;

# Constraints
s.t. constraint: x1 + x2 <=6;

end;
```

And here is the `glpsol` report:

```
Problem:    ip
Rows:       2
Columns:    2 (2 integer, 0 binary)
Non-zeros:  4
Status:     INTEGER OPTIMAL
Objective:  z = 18 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------ ------------- -------------
     1 z                         18
     2 constraint                 6                           6

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------ ------------- -------------
     1 x1           *              6             0
     2 x2           *              0             0

Integer feasibility conditions:

INT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

INT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
```

```
        High quality

End of output
```

Here is the GNU MathProg code for the simple MIP (11.1.2):

```
#
# The MIP (11.1.2)
#

# Decision variables
# Only x1 is required to be integer
var x1 >=0, integer;
var x2 >=0;

# Objective
maximize z: 3*x1 + 2*x2;

# Constraints
s.t. constraint: x1 + x2 <=6;

end;
```

And here is the `glpsol` report:

```
Problem:    mip
Rows:       2
Columns:    2 (1 integer, 0 binary)
Non-zeros:  4
Status:     INTEGER OPTIMAL
Objective:  z = 18 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 z                          18
     2 constraint                  6                           6

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 x1           *              6             0
     2 x2                          0             0

Integer feasibility conditions:

INT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

INT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

End of output
```