Linear Programming with Spice

Or a Gentle Introduction to Linear Models

Tommi Sottinen

tommi.sottinen@uwasa.fi

www.uwasa.fi/∼tsottine/spicy_or/lp_with_spice.pdf

November 28, 2022

Preface

These notes are for the six week 5 ECTS course ORMS1020 Operations Research in the University of Vaasa. I expect that it will take four weeks to go through the material presented here in the course. The first two weeks of the course will contain a short introduction to GNU Octave as presented in the notes *Octave with Spice: Or a Gentle Introduction to GNU Octave Towards Linear Programming* that can be downloaded from `www.uwasa.fi/~tsottine/spicy_or/octave_with_spice.pdf`

These notes are a shortened version of previous notes *ORMS1020 Operations Research with GNU Octave* (2011–09–19) used in 2011–2020. If you are interested, you can download the old notes here: `www.uwasa.fi/~tsottine/or_with_octave/or_with_octave.pdf`

I would like to thank Matti Laaksonen and Rudi Wietsma for carefully reading the manuscript and for pointing out several mistakes.

T.S.
Vaasa November 28, 2022

Contents

Chapter 5

Product-Mix Problem Revisited

I this chapter we revisit the product-mix problem of Muad'Dib bakery that we studied in Chapter 4 of the notes Octave with Spice: Or a Gentle Introduction to GNU Octave Towards Linear Programming. The problem is the same as there. The solution should be the same also. Indeed, they are, but we present the problem and the solution slightly differently here.

Modeling Product-Mix Problem Revisited

5.1 Problem (Muad'Dib Bakery Revisited)
Muad'Dib Bakery produces three types of Fremen cakes: Atreides, Corrino and Harkonnen. Each cake is made out of nutrient powders that are: fat, sugar, protein, water and spice Melange. The composition and selling prices of the cakes and the available nutriet powders are explained in the table below:

	Fremen cake type			
Nutrient powder	Atreides	Corrino	Harkonnen	**Availability**
Fat	70 g	50 g	150 g	100 kg
Sugar	500 g	300 g	500 g	400 kg
Protein	280 g	180 g	50 g	150 kg
Water	600 ml	800 ml	500 ml	300 l
Spice Melange	8 mg	35 mg	10 mg	10 g
Selling Price	120 sol	170 sol	100 sol	

Muad'Dib Bakery wants to maximize daily revenues. What should Muad'Dib Bakery do?

In modeling optimization problems it is usually a good idea to follow the three-step algorithm given in Chapter 4 of Octave with Spice: Or a Gentle Introduction to GNU Octave Towards Linear Programming. We repeat the algorithm here for the readers' convenience:

5.2 Algorithm (Optimization modeling)

Step 1 Find the **decision variables**, i.e. find out what are the variables whose values you can choose.

Step 2 Find the **objective function**, i.e. find out how your objective to be minimized or maximized depends on the decision variables.

Step 3 Find the **constraints**, i.e. find out the (in)equalities that the decision variables must satisfy. (Don't forget the possible sign constraints!)

The idea of Algorithm 5.2 is that you do not try to do everything at once. Indeed, remember that according to Archbishop Emeritus Desmond Tutu there is only one way to eat an elephant: a bite at a time.

Usually, finding the **decision variables** is the most natural starting point. Hence it is **Step 1**. Mind here that we do not what to do everything at once. So, we do not care if some decisions are possible or not. We simply identify what we can **in principle** decide. In the Muad'Dib Bakery problem 5.1 above this is quite obvious. We can decide the number of each types of cakes we are going to make. Let us denote the decision variables unimaginatively as

$$\begin{aligned} x_1 &= \text{number of Atreides cakes produced daily,} \\ x_2 &= \text{number of Corrino cakes produced daily,} \\ x_3 &= \text{number of Harkonnen cakes produced daily.} \end{aligned}$$

Of course, we could have chosen more descriptive variable names. Whether to use descriptive names or standard names, is mostly a matter of taste. Both choices have good and bad aspects.

In **Step 2** we have to find the **objective function**. If we already know the decision variables, then this step is typically an easy one. Indeed, in Problem 5.1 the objective is to maximize the revenue, which, given the decision variables x_1, x_2 and x_3, can be easily read from the last line of the table:

$$z = 120x_1 + 170x_2 + 100x_3.$$

Again, we could have used a more descriptive name for the objective function, like revenue instead of the standard name z.

Finally, in **Step 3** we have to find out all the **constraints** the problem has. This is typically the step where most of the work is done. Luckily the problem data in 5.1 was given in a tabular form that corresponds nicely to the LP formulation we need. Indeed, the constraints can be read from the lines entitled "Fat", "Sugar", "Protein", "Water", and "Spice Melange". We can, if we wish, write them mathematically as

$$\begin{array}{rcrcrcrl} 70x_1 & + & 50x_2 & + & 150x_3 & \le & 100\,000 & \quad\text{(fat)} \\ 500x_1 & + & 300x_2 & + & 500x_3 & \le & 400\,000 & \quad\text{(sugar)} \\ 280x_1 & + & 180x_2 & + & 50x_3 & \le & 150\,000 & \quad\text{(protein)} \\ 600x_1 & + & 800x_2 & + & 500x_3 & \le & 300\,000 & \quad\text{(water)} \\ 8x_1 & + & 35x_2 & + & 10x_3 & \le & 10\,000 & \quad\text{(spice)} \end{array}$$

Note that we wrote $70x_1$ instead $x_1 \times 70$ g, which would have been more "correct". This way or writing, and removing the units, turns out to be more convenient, as we shall soon see. Note that since we removed the units, we had to scale the units to be the same in each row. Different rows can have different units. For example, it would have been no problem to measure protein in kilograms rather than in grams. So, we can change the (protein) row to

$$0.280x_1 + 0.180x_2 + 0.050x_3 \quad \le \quad 150.$$

Indeed, the inequality is the same.

Remembering the implied sign constraints we have arrived to the LP model for the Muad'Dib Bakery problem 5.1:

$$\begin{array}{llrcrcrclll}
\max z & = & 120x_1 & + & 170x_2 & + & 100x_3 & & & & \text{(revenue)} \\
\text{s.t.} & & 70x_1 & + & 50x_2 & + & 150x_3 & \le & 100\,000 & & \text{(fat)} \\
 & & 500x_1 & + & 300x_2 & + & 500x_3 & \le & 400\,000 & & \text{(sugar)} \\
 & & 280x_1 & + & 180x_2 & + & 50x_3 & \le & 150\,000 & & \text{(protein)} \\
 & & 600x_1 & + & 800x_2 & + & 500x_3 & \le & 300\,000 & & \text{(water)} \\
 & & 8x_1 & + & 35x_2 & + & 10x_3 & \le & 10\,000 & & \text{(spice)} \\
 & & & & & & x_1, x_2, x_3 & \ge & 0 & & \text{(sign constraints)}
\end{array} \tag{5.3}$$

5.4 Remark (LP vs. Table)
You should note how similar the table in Problem 5.1 and the LP in (5.3) are. Basically the only difference is that in Problem 5.1 table the prices of cakes are in the bottom while in the LP (5.3) they are in the top.

Solving Product-Mix Problem with GLPK Revisited

We have modeled Muad'Dib Bakery Problem 5.1 as an LP in (5.3). Now we are going to implement the problem with GNU Octave and solve it by using the LP solver glpk. Since GNU Octave is matrix oriented it makes sense to rewrite (5.3) in matrix form. But nothing could be easier! Indeed, let us define

$$\mathbf{c} = \begin{bmatrix} 120 \\ 170 \\ 100 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 70 & 50 & 150 \\ 500 & 300 & 500 \\ 280 & 180 & 50 \\ 600 & 800 & 500 \\ 8 & 35 & 10 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 100\,000 \\ 400\,000 \\ 150\,000 \\ 300\,000 \\ 10\,000 \end{bmatrix}.$$

Then (5.3) can be written in matrix notation as

$$\begin{array}{lrcl}
\max & z = \mathbf{c}'\mathbf{x} & & \\
\text{s.t.} & \mathbf{Ax} & \le & \mathbf{b}\ . \\
 & \mathbf{x} & \ge & \mathbf{0}
\end{array} \tag{5.5}$$

5.6 Remark (LP in Tabular Form)

The reader is invited to compare the matrix form (5.5) and the table in Problem 5.1. Indeed, by using the matrix $\mathbf{A}$ and the vectors $\mathbf{c}$ and $\mathbf{b}$ we can write the table in Problem 5.1 schematically as

Nutrient powder	**Fremen cake type** $\cdots$	**Availability**
$\vdots$	$\mathbf{A}$	$\mathbf{b}$
Selling Price	$\mathbf{c}'$	

Solving Problem 5.1 (Muad'Dib Bakery Revisited) with GNU Octave and glpk is now relatively straingforward, since we have modeled it as a (standrard form) LP in (5.3).

5.7 Solution (Muad'Dib Bakery Revisited)

Here is the script m-file that solves the Muad'Dib Bakery problem 5.1 (you can download it from www.uwasa.fi/~tsottine/spicy_or/muaddib_revisited.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Muad'Dib Bakery Revisited. (Problem 5.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Prices for the cakes Atreides, Corrino, Harkonnen (Solari)
c(1) = 120;               %% Price for Atreides
c(2) = 170;               %% Price for Corrino
c(3) = 100;               %% Price for Harkonnen

%% Nutrient powder composition of the cakes Atreides, Corrino, Harkonnen
A(1,:) = [ 70  50 150];   %% Fat (g)
A(2,:) = [500 300 500];   %% Sugar (g)
A(3,:) = [280 180  50];   %% Protein (g)
A(4,:) = [600 800 500];   %% Water (ml)
A(5,:) = [  8  35  10];   %% Spice Melange (mg)

%% Available nutrients
b(1) = 100;               %% Fat (kg)
b(2) = 400;               %% Sugar (kg)
b(3) = 150;               %% Protein (kg)
b(4) = 300;               %% Water (l)
b(5) =  10;               %% Spice Melange (g)

%% Unit changes kg -> g l -> ml, and g -> mg.
b = 1000*b;

```

```
33 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34 %% Solution with GLPK
35 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36 
37 ctype = "UUUUU";
38 vtype = "CCC";
39 [x_max, z_max] = glpk(c,A,b, [], [], ctype, vtype, -1);
40 
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42 %% Output: x_max for the optimal decision and z_max for the optimal value.
43 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
44 x_max
45 z_max
46 
47 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48 %% Test result:
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 %%>> muaddib_revisited
51 %%x_max =
52 %%
53 %%    171.2329
54 %%    246.5753
55 %%           0
56 %%
57 %%z_max = 6.2466e+04
```

The code above is slightly different that the solution written in Chapter 4 of Octave with Spice: Or a Gentle Introduction to GNU Octave Towards Linear Programming and the m-file muaddib.m there. The reader is strongly encouraged to review that solution now. Here we only point out some differences to that solution.

Like the code muaddib.m the code muaddib_revisited.m above is split into blocks. The first block, lines 1–6, is just a title. The second block, lines 7–32 sets the data of the problem. The third block 33–40 calculates the actual solution. The fourth block 41–46 prints the solution to Command Window. Finally, in the fifth block there is (in comments) how the solution should look like.

The revisited version muaddib.m differs from the earlier version muaddib_revisited.m in following ways

- In the data section, the problem data c, A and b are built row-by-row instead of giving the vectors and the matrices in single commands. It turns out that this way the vectors c and b become row vectors instead of column vectors. It also turns out that glpk does not care about this at all.
- In solution section the glpk parameter ctype and vtype are defined before calling the function glpk. This difference is very minor, almost no worth of mentioning.

Finally, here is the **solution**: Run the m-file muaddib_revisited.m in the GNU Octave console (make sure that the m-file is in you current directory):

```
>> muaddib_revisited
x_max =

171.2329
246.5753
```

```
0

z_max = 6.2466e+04
```

So, Muad'Dib Bakery should produce 171.239 Atreides cakes, 246.5753 Corrino cakes, and no Harkonnen cakes at all. With this choice the daily revenue is maximized and it is 62 466 solaris.

5.8 Exercise (Corrino Copyright)
The Padishah Emperor Shaddam Corrino IV decreed that Corrino cakes are his exclusive copyright: no-one else is allowed to make them. What should Muad'Dib Bakery in Problem 5.1 do now?

Hint: You might want to modify either the m-file muaddib.m or the m-file muaddib_revisited.m to solve this. Extra glory is given to a student who solves this problem in the quick-and-dirty way by modifying a single number in the m-file.

5.9 Exercise (Fenring Cake)
Muad'Dib Bakery of Problem 5.1 is considering to launch a new type of the: Fenring. This cake would contain 120 g of fat, 450 g of sugar, 250 g of protein, 750 ml of water, and no spice at all. The price for Fenring cake would be 120 solaris. Would it be profitable to sell Fenring cakes?

5.10 Exercise (Corrino Licence)
The Padishah Emperor Shaddam Corrino IV decreed that Corrino cakes are his exclusive copyright. To make Corrino cakes, one must pay royalty (imperialty?). How big can this royalty be so that it would still be profitable for Muad'Dib Bakery in Problem 5.1 to make them?

Hint: The obvious way to solve this is to calculate optimal solutions with different royalty prices in a for-loop. Later, when we study **sensitivity analysis** we will learn a more elegant solution.

5.11 Exercise (Muad'Dib with Promises)
Muad'Dib Bakery of Problem 5.1 has made some promises. It has promised to have at least 3 of each cake types for sale each day and also it has promised to deliver 2 g of spice Melange to the imperial treasury and 1 Atreides cake to the planet Caladan each day. What should Muad'Dib Bakery do now?

Chapter 6

Diet and Nutrient Problems

Diet Problem

The diet problem is arguably the first linear optimization problem (LP) studied. Its history dates back to the 1930s and 1940s. The problem was motivated by the U.S. Army's desire to minimize the cost of feeding its soldiers while still providing a healthy diet. So, the goal of the diet problem is to select a set of foods that will satisfy a set of daily nutritional requirement at minimum cost.

Problem 6.1 below is a diet problem version of the product-mix problem of Muad'Dib bakery (Problem 5.1).

6.1 Problem (Sietch Tabr Diet)
Naib Stilgar must feed his sietch. The people in his sietch require five kind of nutrients: fat, sugar, protein, water and spice Melange. In Arrakis there are only three kinds of food: Atreides cakes, Corrino cakes, and Harkonnen cakes. The nutritional requirements of the sietch, the nutritional contents of the cakes, and the prices of the cakes are listed in the table below:

	Fremen cake type			
Nutrient powder	Atreides	Corrino	Harkonnen	**Requirement**
Fat	70 g	50 g	150 g	100 kg
Sugar	500 g	300 g	500 g	400 kg
Protein	280 g	180 g	50 g	150 kg
Water	600 ml	800 ml	500 ml	300 l
Spice Melange	8 mg	35 mg	10 mg	10 g
Price	120 sol	170 sol	100 sol	

Naib Stilgar wants to feed his sietch at minimum cost. What should Naib Stilgar buy?

The diet problem 6.1 is very close to the product-mix problem 5.1. So, one might think that their solutions are the same, since the Sietch Tabr nutrient requirements are exactly the same as Muad'Dib Bakery's available nutrients. This is not however the case. Indeed, the solutions are not the same, as we shall see soon. However modeling and solving the problem is very similar to the product-mix problem of Muad'Dib bakery 5.1.

To model the diet problem 6.1 we can use the same approach as used in modeling the product-mix problem 5.1. Since the differences are minor, we only give the solution here:

$$
\begin{array}{lrcrcrcrcrl}
\min w & = & 120x_1 & + & 170x_2 & + & 100x_3 & & & & \text{(cost)} \\
\text{s.t.} & & 70x_1 & + & 50x_2 & + & 150x_3 & \geq & 100\,000 & & \text{(fat)} \\
 & & 500x_1 & + & 300x_2 & + & 500x_3 & \geq & 400\,000 & & \text{(sugar)} \\
 & & 280x_1 & + & 180x_2 & + & 50x_3 & \geq & 150\,000 & & \text{(protein)} \\
 & & 600x_1 & + & 800x_2 & + & 500x_3 & \geq & 300\,000 & & \text{(water)} \\
 & & 8x_1 & + & 35x_2 & + & 10x_3 & \geq & 10\,000 & & \text{(spice)} \\
 & & & & & & x_1, x_2, x_3 & \geq & 0 & & \text{(sign constraints)}
\end{array}
\tag{6.2}
$$

The reader is encouraged to compare the LP model (6.2) to the LP model (5.3).

6.3 Solution (Sietch Tabr Diet)

Here is the script m-file that solves the Sietch Tabr Diet Problem 6.1 (you can download it from www.uwasa.fi/~tsottine/spicy_or/tabr_diet.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Sietch Tabr Diet. (Problem 6.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Prices for Atreides, Corrino, Harkonnen (Solari)
c = [120 170 100]';

%% Nutrient powder composition of the cakes Atreides, Corrino, Harkonnen
A = [ 70  50 150;                %% Fat (g)
     500 300 500;                %% Sugar (g)
     280 180  50;                %% Protein (g)
     600 800 500;                %% Water (ml)
       8  35  10 ];              %% Spice Melange (mg)

%% Required nutrients (note unit scale)
b = 1000*[100 400 150 300 10]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solution with GLPK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[x, w] = glpk(c,A,b, [], [], "LLLLL", "CCC", 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Output: x for the optimal decision and w for the optimal value.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x, w

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
36 %% Test resulta tabr_diet vs. muaddib
37 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38
39 %%>> tabr_diet
40 %%x =
41 %%
42 %%    415.385
43 %%     61.538
44 %%    452.308
45 %%
46 %%w = 1.0554e+05
47
48 %%>> muaddib
49 %%x_max =
50 %%
51 %%    171.2329
52 %%    246.5753
53 %%           0
54 %%
55 %%z_max = 6.2466e+04
```

The reader should note that the code tabr_diet.m is almost identical to the code muaddib.m (see www.uwasa.fi/∼tsottine/spicy_or/muaddib.m). The only essential differences are in the line 28.

- In the diet problem the constraints are of type ≥. Therefore the ctype input paramer is "LLLLL" instead of "UUUUU".
- In the diet problem we are minimizing instead of maximizing. Therefore the sense input parameter is 1 instead of −1. (Since minimization is default for glpk, we could have simply omitted this input parameter).

The solution is given in the comments in the lines 39–46: Naib Stilgar should buy 415.3859 Atreides cakes, 61.538 Corrino cakes, and 452.308 Harkonnen cakes. The total cost of the cakes is 105 538.46 solaris.

Finally, in the lines 48–55 there is in the comments the solution of Muad'Dib product-mix problem 5.1 for comparison.

6.4 Exercise (Muad'Dib Needs More Nutrients)
Note that Muad'Dib Bakery of Problem 5.1 does not have enough daily nutrients to feed Sietch Tabr of Problem 6.1. How much nutrients should Muad'Dib bakery have in order to be able meet Sietch Tabr's needs?

6.5 Exercise (Buy–Sell Comparison Theorem)
This exercise was wrong. Sorry for the confusion!

Nutrient Problem

The product-mix problem 5.1 and the diet problem 6.1 were closely related but somehow not in balance: while the Muad'Dib bakery has all the nutrients Sietch Tabr needs, it cannot provide for the sietch. The reason is that Muad'Dib bakery sells fremen cakes, not nutrient powders and some of the nutrients remain unused. If Muad'Dib bakery allows to directly buy the powders, the situation changes.

6.6 Problem (Sietch Tabr Nutrient Diet)
Naib Stilgar must feed his sietch. The people in his sietch require five kind of nutrients: fat, sugar, protein, water and spice Melange. The requirements of the sietch are

Nutrient powder	**Requirement**
Fat	100 kg
Sugar	400 kg
Protein	150 kg
Water	300 l
Spice Melange	10 g

Naib Stilgar wants to buy the nutrient powders from Muad'Dib bakery. What should Naib Stilgar's offer to Muad'Dib bakery be?

Let us model Problem 6.6 by using Algorithm 5.2.

In **Step 1** we should find out the **decision variables**. So what can Naib Stilgar decide? He wants to buy nutrient powders to feed his sietch. So, the decision variables are the offered prices (per unit) for the nutrient powders. Let us call these prices $\mathbf{y} = [y_1\ y_2\ y_3\ y_4\ y_5]'$. So, $\mathbf{x}$ will denote the solution for Muad'Dib bakery Problem 5.1 and $\mathbf{y}$ will denote the solution for Naib Stilgar's Problem 6.6. So, we have the decision variables

$$\begin{aligned}
y_1 &= \text{price (in solaris) for 1 g of fat powder,}\\
y_2 &= \text{price (in solaris) for 1 g of sugar powder,}\\
y_3 &= \text{price (in solaris) for 1 g of protein powder,}\\
y_4 &= \text{price (in solaris) for 1 ml of water powder,}\\
y_5 &= \text{price (in solaris) for 1 mg of spice Melange powder.}
\end{aligned}$$

In **Step 2** we need to find out the **objective function**. So what is Naib Stilgar's objective? It is obviously to feed his sietch with minimal cost. So, this is a minimization problem. Therefore we call the objective function w, and for comparison z will denote the objective for Muad'Dib's Problem 5.1. Since the nutrient requirements are given in the problem data and the price per unit for the nutrients are the decision variables, the objective function to be minimized is

$$w = 100\,000\, y_1 + 400\,000\, y_2 + 150\,000\, y_3 + 300\,000\, y_4 + 10\,000\, y_5$$

In **Step 3** we need to find out the **constraints**. To figure out what they are we have to think what kind of offers would Muad'Dib Bakery accept. Suppose Naib Stilgar gives an offer

$\mathbf{y} = [y_1\ y_2\ y_3\ y_4\ y_5]'$ per unit for nutrients to buy **all** the nutriets Muad'Dib Bakery has. Now Muad'Dib bakery has two choices: either accept the offer and sell all its nutrients or refuse the offer and use its nutrients to make fremen cakes. Recall what the Muad'Dib's business is:

	Fremen cake type			
Nutrient powder	Atreides	Corrino	Harkonnen	**Availability**
Fat	70 g	50 g	150 g	100 kg
Sugar	500 g	300 g	500 g	400 kg
Protein	280 g	180 g	50 g	150 kg
Water	600 ml	800 ml	500 ml	300 l
Spice Melange	8 mg	35 mg	10 mg	10 g
Selling Price	120 sol	170 sol	100 sol	

Let us consider the first column "Atreides". This column tells that Muad'Dib Bakery can turn 70 g of fat, 500 g of sugar, 280 g of protein, 600 ml of water, and 8 mg of spice into a cake that sells for 120 solaris. Now, suppose that Naib Stilgar's offer $\mathbf{y}$ is such that

$$70y_1 + 500y_2 + 280y_3 + 600y_4 + 8y_5 \geq 120. \tag{6.7}$$

This means that Naib Stilgar is offering to buy nutrients in a better price than Muad'Dib Bakery could get from baking them into Atreides cakes. Suppose Naib Stilgar's offer is better for Corrino and Harkonnen cakes also, i.e., we have in addition to (6.7)

$$50y_1 + 300y_2 + 180y_3 + 800y_4 + 35y_5 \geq 170, \tag{6.8}$$

$$150y_1 + 500y_2 + 50y_3 + 500y_4 + 10y_5 \geq 100. \tag{6.9}$$

Inequalities (6.7), (6.8), and (6.9) mean that Naib Stilgar's offer is better than making the cakes. So, Muad'Dib Bakery should accept the offer. On the other hand, if any of the constraints (6.7), (6.8), or (6.9) is not satisfied, then it would be better for Muad'Dib Bakery to make some cakes instead of selling all its nutrients to Naib Stilgar. This means that Step 3 is finished. We have found out the constraints, and we can collect collect what we have found out as an LP:

$$\begin{array}{lrcrcrcrcrcr}
\min w = & 100\text{k}y_1 & + & 400\text{k}y_2 & + & 150\text{k}y_3 & + & 300\text{k}y_4 & + & 10\text{k}y_5 & & \\
\text{s.t.} & 70y_1 & + & 500y_2 & + & 280y_3 & + & 600y_4 & + & 8y_5 & \geq & 120 \\
 & 50y_1 & + & 300y_2 & + & 180y_3 & + & 800y_4 & + & 35y_5 & \geq & 170 \\
 & 150y_1 & + & 500y_2 & + & 50y_3 & + & 500y_4 & + & 10y_5 & \geq & 100 \\
 & & & & & & & & & y_1, y_2, y_3, y_4, y_5 & \geq & 0
\end{array} \tag{6.10}$$

We used kilo units in the objective line. The reason for this was marginal.

6.11 Remark (Bakery–Sietch Duality)
We have seen our first **duality**. Indeed, consider the Muad'Dib table

	Fremen cake type			
Nutrient powder	Atreides	Corrino	Harkonnen	**Availability**
Fat	70 g	50 g	150 g	100 kg
Sugar	500 g	300 g	500 g	400 kg
Protein	280 g	180 g	50 g	150 kg
Water	600 ml	800 ml	500 ml	300 l
Spice Melange	8 mg	35 mg	10 mg	10 g
Selling Price	120 sol	170 sol	100 sol	

Recall that setting

$$\mathbf{c} = \begin{bmatrix} 120 \\ 170 \\ 100 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 70 & 50 & 150 \\ 500 & 300 & 500 \\ 280 & 180 & 50 \\ 600 & 800 & 500 \\ 8 & 35 & 10 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 100\ 000 \\ 400\ 000 \\ 150\ 000 \\ 300\ 000 \\ 10\ 000 \end{bmatrix}$$

we can write the table above formally as

Nutrient powder	**Fremen cake type** $\cdots$	**Availability**
$\vdots$	$\mathbf{A}$	$\mathbf{b}$
Selling price	$\mathbf{c}'$	

and the Muad'Dib product-mix problem can be written as an LP as

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

Now, the dual way of looking at the Fremen cake type table is to read it row-wise instead of column-wise (or the other way around). This means that we are looking at the transposed table

Fremen cake type	**Nutrient powder** $\cdots$	**Selling price**
$\vdots$	$\mathbf{A}'$	$\mathbf{c}$
Requirement	$\mathbf{b}'$	

and the Naib Stilgar's nutrient purchase problem can be written as an LP as

$$\begin{array}{lrcl} \min & w = \mathbf{b}'\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}'\mathbf{y} & \geq & \mathbf{c} \\ & \mathbf{y} & \geq & \mathbf{0} \end{array} .$$

6.12 Exercise (Sietch Tabr Nutrient Diet)
Solve the Nutrient Diet Problem 6.6 by using glpk. Compare the solution with the Muad'Dib Bakery product-mix problem solution 5.7.

Chapter 7

Sensitivity

The optimal value of (a standard form) LP

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

can be considered as a function of its parameters (or arguments, if you like) $\mathbf{c}$, $\mathbf{A}$, and $\mathbf{b}$:

$$z^* = z^*(\mathbf{c}, \mathbf{A}, \mathbf{b}).$$

Of course, this function is a black box to us. We no not know how e.g. glpk calculates it. But it is a function, nevertheless.

As for any function, for function z^*, it is important to understand how it changes when its parameters change. These changes are of course the partial derivatives

$$\begin{array}{cl} \dfrac{\partial z^*}{\partial c_i} & \text{for the changes in the objective function,} \\ \dfrac{\partial z^*}{\partial A_{i,j}} & \text{for the changes in the technology coefficient,} \\ \dfrac{\partial z^*}{\partial b_j} & \text{for the changes in the available resources.} \end{array}$$

The changes $\partial z^*/\partial c_i$ or $\partial z^*/\partial A_{ij}$ have no standard name, and we shall not consider them.

The changes $\partial z^*/\partial b_j$ are called the **shadow prices** and they are usually denoted by λ_j's (or μ_j's or π_j's).

The so-called **reduced costs** are related to the changes in the objective function, but they are **not the partial derivatives** with respect to $\mathbf{c}$. Instead, they are related to the decision variables x_i we decide not to produce, i.e., $x_i^* = 0$. So, one could say that the reduced costs are related also to the optimal decision

$$\mathbf{x}^* = \mathbf{x}^*(\mathbf{c}, \mathbf{A}, \mathbf{b}).$$

However, they are **not** the partial derivatives $\partial \mathbf{x}^*/\partial c_i$ either (whatever does that even mean).

Shadow Prices a.k.a. Dual Variables a.k.a. Marginal Prices

In this section we are interested in the changes

$$\lambda_j \;=\; \frac{\partial z^*}{\partial b_j}$$

for an optimal solution $z^* = z^*(\mathbf{c}, \mathbf{A}, \mathbf{b})$. So, we have a change in one of the resources $\mathbf{b}$.

7.1 Problem (Muad'Dib's Jihad)
Having won the war against the Corrino imperial dynasty, the new emperor Paul Muad'Dib of the house Atreides needs to consolidate his power. Paul Muad'Dib wants to build a rapid deployment force that can be sent to different planets if needed.

Paul Muad'Dib has unlimited supply of three kinds of forces: the zealous elite force fedaykin, standard Atreides conscripts, and deserted sardaukar troops. In consolidating the new imperial power, the relative efficiency for fedaykin, conscripts, and sardaukar are 10, 1, and 7, respectively. Thus, if the rapid deployment force has

$$\begin{aligned} x_1 &= \text{“fedaykin legions”}, \\ x_2 &= \text{“conscript legions”}, \\ x_3 &= \text{“sardaukar legions”}, \end{aligned}$$

then the influence of the rapid deployment force is

$$z \;=\; 10x_1 + x_2 + 7x_3.$$

Depending on the planet where the rapid deployment force may be deployed, some troops will generate more resentment than other troops. Also, sometimes the troops are welcome, i.e., they generate negative resentment. Finally, the level of tolerance for resentment is different for different planets. If the tolerance level is exceeded, the planet will rebel. The tolerance to resentment and resentment generated by different types of legions of troops is presented in the table below:

	Resentment			
Planet	Fedaykin	Conscript	Sardaukar	**Tolerance**
Arrakis	-1	0	2	89
Caladan	0	-1	1	780
Chapterhouse	-1	1	1	68
Giedi Prime	2	1	0	740
Ix	1	0	1	599
Kaitain	3	2	-1	1250
Tleilax	1	1	1	595
Influence	10	1	7	

So, for example, people in Arrakis like the Fedaykin, don't care about the conscripts, but hate the sardaukar, while people in Tleilax dislike all the troops equally.

Emperor Paul Muad'Dib wants to maximize the influence of his rapid deployment force and in the same time keep the planets from rebelling because of resentment to the rapid deployment force.

To increase the resentment tolerance, Paul Muad'Dib can bestow one planet "special imperial" status. This does not cost anything (since it is a purely ceremonial status), but increase that planet's tolerance by 10 units. Which planet should have the "special imperial" status?

7.2 Remark (Jihad Differential)

Let $z^* = z^*(\mathbf{c}, \mathbf{A}, \mathbf{b})$ be the solution of the Muad'Dib Jihad's LP of Problem 7.1. That is, $\mathbf{c}$ is influence to be optimized, $\mathbf{A}$ is the technology (tolerance) matrix, $\mathbf{b}$ is the tolerance levels, and z^* is the optimal influence.

By using vector notation , the "special imperial" status should be given to the planet i $(i = 1, 2, 3, 4, 5, 6, 7)$ for which

$$z^*(\mathbf{c}, \mathbf{A}, \mathbf{b} + 10\mathbf{e}_i) \tag{7.3}$$

is the greatest. Here $\mathbf{e}_i$ is the 7-dimensional vector having 1 at the ith element and 0 otherwise.

A safe way of solving Problem 7.1 is to calculate the optimal influences of (7.3) for all the 7 different choices and then to choose the best one. This is left as an exercise.

An unsafe, but more informative, way of solving Problem 7.1 is to check the **shadow prices** of the original LP solution $(z^*, \mathbf{x}^*)$, where $z^* = z^*(\mathbf{c}, \mathbf{A}, \mathbf{b})$. This is what we do next.

7.4 Solution (Muad'Dib's Jihad)

The following m-file, jihad.m solves Problem 7.1. It can be downloaded by using the link www.uwasa.fi/~tsottine/spicy_or/jihad.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Muad'Dib's Jihad. (Problem 7.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Influence from fedaykin, conscripts, and sardaukar.
c = [
10;                                     %% Feadykin
 1;                                     %% Conscripts
 7;                                     %% Sardaukar
];

%% Resentment tolerancies for the planets:
```

```
b = [
  89;                                 %% Arrakis
 780;                                 %% Caladan
  68;                                 %% Chapterhouse
 740;                                 %% Giedi Prime
 599;                                 %% Ix
1250;                                 %% Kaitain
 595;                                 %% Tleilax
];

%% Tolerance matrix (fedaykin, conscript, sardaukar) for the planets
A = [
        -1          0          2;     %% Arrakis
         0         -1          1;     %% Caladan
        -1          1          1;     %% Chapterhouse
         2          1          0;     %% Giedi Prime
         1          0          1;     %% Ix
         3          2         -1;     %% Kaitain
         1          1          1;     %% Tleilax
];
%% Fedaykin  Conscripts  Sardaukar

%% Set constraints "U" for all the planets
ctype = "";
for i=1:length(b)
    ctype = [ctype "U"];
end

%% Set variable type "C" for all the troop types
vtype = "";
for i=1:length(c)
    vtype = [vtype "C"];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solution with GLPK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[x, z, e, xtra] = glpk(c,A,b, [], [], ctype, vtype, -1);
lambda = xtra.lambda;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Output: All the vectors are transposed for "compact" view.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

optimal_legions = x'
optimal_influence = z'
marginal_influence_gains = lambda'
free_tolerances = (b-A*x)'

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test result:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%>> jihad
%%optimal_legions =
%%
```

```
76 %%    370       0     225
77 %%
78 %%optimal_influence = 5275
79 %%marginal_influence_gains =
80 %%
81 %%          0          0          0    1.5000          0          0    7.0000
82 %%
83 %%free_tolerances =
84 %%
85 %%      9    555    213      0      4    365      0
```

The m-file jihad.m works pretty much the same way as our previous m-files. There are however a couple of new fine points that are worth mentioning:

- In lines 47–51 the glpk's input parameters ctype and vtype are set in a dynamic way so that the glpk call in the line 57 will work even if you change the dimensions of the problem data **c**, **A**, **b**. That means that adding or removing planets and troop types will work just fine for the rest of the file: the data section and the solution section are now truly independent.
- In line 57 we ask glpk to give us the full answer. A particularly interesting additional part in the full answer is the field xtra.lambda, which we decided to call simply lambda ($\boldsymbol{\lambda}$) in the line 58.
- Finally, in the output section we decided to give the output parameters descriptive names and transposed them for more convenient reading. Finally, we also added a new output vector free_tolerances that tells us how much tolerance there is left in the planets if they are occupied by the rapid deployment force.

Finally, the **solution** is to grant the planet Tleilax the "special imperial" status. Indeed, that would (if the change of 10 units is small) increase the influence of the rapid deployment force by $7 \times 10 = 70$ units. For comparison, giving the "special imperial" status to the planet Giedi Prime would increase the rapid deployment force's influence by $1.5 \times 10 = 15$ units. All other choices would not increase the influence at all for the obvious reason: their tolerances are not fully used, so increasing them should not change anything.

7.5 Exercise (Safe Jihad Policy)

Solve the "special imperial" bestowing selection of Problem 7.1 in the safe way by solving all the 7 different LP's (7.3) as explained in Remark 7.2.

7.6 Exercise (Tleilax Rebellion)

The planet Tleilax decided to rebel against the just and holy rule of Emperor Paul Muad'Dib. Therefore a permanent garrison was stationed there. So, there is no point in sending rapid deployment forces nor granting Tleilax the "special imperial" status.

In this new situation, which planet should be bestowed the "special imperial" status in Problem 7.1?

7.7 Exercise (Muad'Dib Bakery's Nutrient Purchase)
Recall Muad'Dib Bakery from Problem 5.1. Suppose Muad'Dib Bakery is given the opportunity to buy the nutrient powders from the notorious Sietch Jacurutu. How much should Muad'Dib Bakery be willing to pay for each of the nutrient powders?

Let us end this section with a discussion on **what and why** the shadow prices are.

7.8 Remark (On Shadow Prices, What and Why)

- Shadow prices are the partial derivatives $\partial z^*/\partial b_j$. They are also denoted by λ_j. (Sometimes π_j or μ_j)
- Shadow prices are called **shadow prices**, since they are the optimal solution of a **shadow problem**, the so called the **dual problem**. We will learn about this later. This is also the reason why shadow prices are also called **dual variables**.
- Shadow prices are also called **marginal prices**. This means that they tell the price of the resources **in the margin**: one unit of increase in resource b_j will increase the (previous) **optimal solution** z^* by λ_j units, **if the change of one unit is "small"**.
- Actually, the function z^* is linear for "small" changes (but not for "big" ones). This means that if $\boldsymbol{\delta} = [\delta_1 \ \cdots \ \delta_n]$ is "small" then the differential approximation is not an approximation at all, but an equality:

$$\begin{aligned} z^*(\mathbf{c},\mathbf{A},\mathbf{b}+\boldsymbol{\delta}) &= z^*(\mathbf{c},\mathbf{A},\mathbf{b}) + \frac{\partial z^*}{\partial \mathbf{b}}(\mathbf{c},\mathbf{A},\mathbf{b})'\boldsymbol{\delta} \\ &= z^*(\mathbf{c},\mathbf{A},\mathbf{b}) + \boldsymbol{\lambda}'\boldsymbol{\delta}, \end{aligned} \tag{7.9}$$

  where we have used the vector gradient notation:

$$\frac{\partial z^*}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial z^*}{\partial b_1} \\ \frac{\partial z^*}{\partial b_2} \\ \vdots \\ \frac{\partial z^*}{\partial b_m} \end{bmatrix}.$$

  Equation (7.9) gives a compact answer to the question **why** for shadow prices (and also to the **what**). If the changes $\boldsymbol{\delta}$ are "small", we can see the changes in the optimal solution neatly without any reason to calculate huge number of LP's for all the changes.
- If a shadow price $\lambda_j = 0$, this **usually** means that not all the resources b_j are used in the optimal solution. In other words

$$A_{j1}x_1^* + \cdots + A_{jn}x_n^* \;<\; b_j,$$

  where $\mathbf{x}^* = [x_1 \ \cdots \ x_n]'$ is the optimal solution of the (standard form) LP.

Reduced Costs a.k.a. Opportunity Costs

In this section we are interested in the changes in the objective parameter $\mathbf{c}$ for both on the **optimal decision** $\mathbf{x}^* = \mathbf{x}^*(\mathbf{c},\mathbf{A},\mathbf{b})$, and on the optimal value $z^* = z^*(\mathbf{c},\mathbf{A},\mathbf{b})$.

The slightly complicated definition for the reduced cost is as follows: The **reduced cost** u_i associated with the coefficient c_i is zero if $x_i^* \neq 0$. If $x_i^* = 0$, then u_i is the amount c_i has to improve (reduce for minimization and increase for maximization) to make x_i^* non-zero.

7.10 Problem (Bene Gesserit Conscript Training)
The optimal solution to Problem 7.1 was to compose the rapid deployment force with

$$\begin{aligned} x_1^* &= 370 && \text{legions of fedaykin zealots,} \\ x_2^* &= 0 && \text{legions of Atreides conscripts,} \\ x_3^* &= 225 && \text{legions of sardaukar deserters} \end{aligned}$$

(see Solution 7.4). This means that the Atreides conscripts are not influential enough to be included in the rapid deployment force.

The Bene Gesserit have promised to teach the Atreides conscripts the Weirding Way fighting technique. This should make them more influential. Of course, the Bene Gesserit demand a price for their services. How much should the Weirding Way increase the Atreides conscripts influence before any negotiations of the price with the Bene Gesserit makes any sense?

7.11 Remark (Bene Gesserit Training Reduced Cost)
Let $z^* = z^*(\mathbf{c},\mathbf{A},\mathbf{b})$ be the solution of the Muad'Dib Jihad's LP of Problem 7.1. That is, $\mathbf{c}$ is influence to be optimized, $\mathbf{A}$ is the technology (tolerance) matrix, $\mathbf{b}$ is the tolerance levels, and z^* is the optimal influence.

By using vector notation, the Bene Gesserit training for the Atreides conscripts ($i = 2$) would increase the influence to

$$z^*(\mathbf{c}+\delta_2\mathbf{e}_2,\mathbf{A},\mathbf{b}). \tag{7.12}$$

In order for this training to make sense for Emperor Paul Muad'Dib, it must be so that

$$z^*(\mathbf{c}+\delta_2\mathbf{e}_2,\mathbf{A},\mathbf{b}) \;>\; z^*(\mathbf{c},\mathbf{A},\mathbf{b}), \tag{7.13}$$

and, consequently, $x_2^* > 0$.

As safe but tedious way to solve Problem 7.10 is to calculate (7.12) in a **for** loop for δ_2 to find out the breaking point, where (7.13) holds.

An unsafe, but more informative, way of solving Problem 7.1 is to check the **reduced costs** of the original LP solution $(z^*, \mathbf{x}^*)$, where $z^* = z^*(\mathbf{c},\mathbf{A},\mathbf{b})$. This is what we do next.

7.14 Solution (Bene Gesserit Conscript Training)

The unsafe solution is of course to look at the reduced cost of the original solution of the Jihad problem presented in jihad.m. Here is a quick-and-dirty modification of jihad.m, called consript.m, downloadable from www.uwasa.fi/~tsottine/spicy_or/conscript.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Bene Gesserit Atreides Conscript Training
%% (Problem 7.10 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Influence from fedaykin, conscripts, and sardaukar.
c = [
10;                                     %% Feadykin
 1;                                     %% Conscripts
 7;                                     %% Sardaukar
];

%% Resentment tolerancies for the planets:
b = [
  89;                                   %% Arrakis
 780;                                   %% Caladan
  68;                                   %% Chapterhouse
 740;                                   %% Giedi Prime
 599;                                   %% Ix
1250;                                   %% Kaitain
 595;                                   %% Tleilax
];

%% Tolerance matrix (fedaykin, conscript, sardaukar) for the planets
A = [
        -1          0           2;      %% Arrakis
         0         -1           1;      %% Caladan
        -1          1           1;      %% Chapterhouse
         2          1           0;      %% Giedi Prime
         1          0           1;      %% Ix
         3          2          -1;      %% Kaitain
         1          1           1;      %% Tleilax
];
%% Fedaykin   Conscripts   Sardaukar

%% Set constraints "U" for all the planets
ctype = "";
for i=1:length(b)
    ctype = [ctype "U"];
end

%% Set variable type "C" for all the troop types
vtype = "";
for i=1:length(c)
    vtype = [vtype "C"];
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solution with GLPK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[x, z, e, xtra] = glpk(c,A,b, [], [], ctype, vtype, -1);
u = xtra.redcosts;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Output: Reduced costs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test output:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%>> conscript
%%u =
%%
%%          0
%%    -7.5000
%%          0
```

What does this mean? Apparently, the **solution** is that the Atreides conscript influence should be increased by 7.5 points (never mind the sign). Since the original efficiency was 1, this means that the new atreides influence should be $c_2 = 1 + 7.5 = 8.5$.

7.15 Exercise
Find the breaking point (7.13) for the Bene Gesserit Weirding Way training gain δ_2 for the Atreides conscripts as explained in Remark 7.11 by running the parameter δ_2 with small increments in a **for** loop.

Is the breaking point solution you found the same as in Solution 7.14? If yes, then why? If not, then why?

7.16 Exercise
Recall the Muad'Dib Bakery Problem 5.1. In Solution 5.7 it turned out that Harkonnen cakes are not produced at all. So, their price is too low. What would be the "correct" price for the Harkonnen cakes so that it would make sense for the Muad'Dib bakery to make them?

Let us end this section, and the whole chapter, with a discussion on **what and why** the reduced costs are.

7.17 Remark (On Reduced Costs, What and Why)

- Reduced costs are related to the objective function coefficients c_i. They are usually denoted by u_i.
- Reduced costs, like the shadow prices, are also related to the **shadow problem**, a.k.a. the **dual problem**. We will later learn how.
- Reduced costs are also called **opportunity costs**. 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 u_i.

Chapter 8

Duality

In this chapter we consider LP's rather formally, or theoretically, if you like. We start in the first section "Standard Form, Dual Standard Form, and Slack Form" with a rather lengthy study on how to present LP's in different but equivalent forms. The actual duality between LP's is then given in the second section "Duality Theorems".

Standard Form, Dual Standard Form, and Slack Form

An LP is in **standard form** if it is of the form

$$\begin{array}{lrcl} \max & z = \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \le & \mathbf{b} \\ & \mathbf{x} & \ge & \mathbf{0} \end{array} . \tag{8.1}$$

We have used, even assumed, this standard form many times previously. So, it seems that we have restricted our considerations only to standard form LP's and it could be the case that our analysis is not valid for all LP's. Fortunately, this is not the case. Indeed, **any LP can be written in a standard form**.

In general, LP's can take many different forms. Also, we have not even given a formal definition of an LP. We continue not to do so. Instead, we proceed with problems and examples.

Let us start with a simple example.

8.2 Example (Simple Standard LP Transformation)
Consider the non-standard form LP

$$\begin{array}{lrrrrrrrrl} \min z & = & 8x_1 & - & 10x_2 & + & 7x_3 & & & \text{(I)} \\ \text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3 & \ge & 3 & \text{(II)} \\ & & x_1 & & & + & 6x_3 & \le & 42 & \text{(III)} \\ & & & & & & x_1, x_2, x_3 & \ge & 0 & \text{(IV)} \end{array} \tag{8.3}$$

Consider the objective (I) in (8.3). It is a minimization. But to minimize z is to maximize $-z$. So, we can simply multiply the objective by -1, and our objective becomes

$$\max \quad -z \;=\; -8x_1 \;+\; 10x_2 \;-\; 7x_3 \qquad \text{(I')}$$

Consider then the inequality (II) in (8.3). Multiply the inequality by -1, and you see that (II) is the same as

$$-9x_1 \;-\; 5x_2 \;+\; 3x_3 \;\leq\; -3 \qquad \text{(II')}$$

Finally, the lines (III) and (IV) look just fine for standard form purposes. Therefore we have found out that the LP (8.3) is the same as the following LP:

$$\text{(8.4)}\qquad \begin{array}{lrcrcrcrcrl} \max & -z & = & -8x_1 & + & 10x_2 & - & 7x_3 & & & \text{(I')} \\ \text{s.t.} & & & -9x_1 & - & 5x_2 & + & 3x_3 & \leq & -3 & \text{(II')} \\ & & & x_1 & & & + & 6x_3 & \leq & 42 & \text{(III)} \\ & & & & & & & x_1, x_2, x_3 & \geq & 0 & \text{(IV)} \end{array} .$$

8.5 Exercise (Standard and Non-Standard LP's with GLPK)
Solve both the non-standard form LP (8.3) and the standard form LP (8.4) with glpk. Note that someting will go wrong with glpk! To understand what, happens ask for the full outputs [x_opt, z_opt, errnro, xtra]. You might also want to tune the workings of glpk by setting msglev to the maximum and turn the presolver off.

Let us then consider an LP that is pretty far from a standard form.

8.6 Problem (Non-Standard LP)
Consider the LP

$$\text{(8.7)}\qquad \begin{array}{lrcrcrcrcrl} \min z & = & & 11x_1 & - & 10x_2 & + & 7x_3 & & & \text{(I)} \\ \text{s.t.} & & & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 & \text{(II)} \\ & 2 & \leq & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III)} \\ & & & x_1 & - & 4x_2 & + & 8x_3 & = & 50 & \text{(IV)} \\ & & & & & & & x_1, x_3 & \geq & 0 & \text{(V)} \\ & & & & & & & x_2 & & \text{urs} & \text{(VI)} \end{array}$$

The LP (8.7) is obviously not in standard form (8.1). Our task is to write it in standard form.

8.8 Remark (urs)
In (8.7) the final "constraint" (VI) is redundant: urs is short for "unrestricted in sign". Since the sign constraints are so typical, it is sometimes a good idea to emphasize when we don't have them.

To write an LP like the one in Problem 8.6 in standard form, one can follow Algorithm 8.10 given below. All the other parts of Algorithm 8.10 are probably quite obvious except Step 4 where the urs (unrestricted in sign) variables are split into "restricted in sign" variables. The next remark elaborates this point.

8.9 Remark (Positive and Negative Part)
Let x be any real number. We can always split x into its positive and negative part as

$$x \; = \; x^+ - x^-,$$

where both x^+ and x^- are positive (i.e. non-negative). Indeed, we can set

$$\begin{aligned} x^+ &= \max(x,0), \\ x^- &= \max(-x,0). \end{aligned}$$

For example

$$\begin{aligned} 42 &= \max(42,0) - \max(-42,0) \\ &= 42 - 0, \\ -7 &= \max(-7,0) - \max(-(-7),0) \\ &= 0 - \max(7,0) \\ &= 0 - 7. \end{aligned}$$

8.10 Algorithm (LP Standard Form Algorithm)

Step 1 Change into maximization: if the objective is to maximize, do nothing. If the objective is to minimize, change it to maximize by multiplying the objective by -1:

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

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

$$\begin{aligned} & l_i \leq \; A_{i1}x_1 + \cdots + A_{in}x_n \; \leq u_i \\ \rightsquigarrow & \left\{ \begin{array}{lll} l_i \leq & A_{i1}x_1 + \cdots + A_{in}x_n & \\ & A_{i1}x_1 + \cdots + A_{in}x_n & \leq u_i \end{array} \right. . \end{aligned}$$

Note that and equality is actually a double inequality. So, transform

$$\begin{aligned} & A_{i1}x_1 + \cdots + A_{in}x_n \; = \; b_i \\ = \; & b_i \leq \; A_{i1}x_1 + \cdots + A_{in}x_n \leq \; b_i \\ \rightsquigarrow & \left\{ \begin{array}{lll} b_i \leq & A_{i1}x_1 + \cdots + A_{in}x_n & \\ & A_{i1}x_1 + \cdots + A_{in}x_n & \leq b_i \end{array} \right. . \end{aligned}$$

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:

$$l_i \; \leq \; A_{i1}x_1 + \cdots + A_{in}x_n \quad \rightsquigarrow \quad -A_{i1}x_1 - \cdots - A_{in}x_n \; \leq \; -l_i.$$

Step 4 Impose sign constraints by splitting the decision variables: If the decision variable x_i is not restricted in sign to be positive (or is urs), then replace it everywhere with $x_i = x_i^+ - x_i^-$ where $x_i^+, x_i^- \geq 0$ are now restricted in sign to be positive.

8.11 Solution (Non-Standard LP)
Let us transform the LP (8.7) of Problem 8.6 into a standard form LP by using Algorithm 8.10. So, we are dealing with the non-standard form LP

$$\begin{array}{lrrrrrrrl}
\min z = & & 11x_1 & - & 10x_2 & + & 7x_3 & & & \text{(I)}\\
\text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 & \text{(II)}\\
 & 2 \leq & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III)}\\
 & & x_1 & - & 4x_2 & + & 8x_3 & = & 50 & \text{(IV)}\\
 & & & & & & x_1, x_3 & \geq & 0 & \text{(V)}\\
 & & & & & & x_2 & & \text{urs} & \text{(VI)}
\end{array}$$

We start with **Step 1**. Since our LP is not a maximization, we turn it into a maximization by multiplying the objective row (I) by -1. Thus, our LP is now in the form

$$\begin{array}{lrrrrrrrll}
\max\ -z = & & -11x_1 & + & 10x_2 & - & 7x_3 & & & \text{(I')}\\
\text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 & \text{(II)}\\
 & 2 \leq & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III)}\\
 & & x_1 & - & 4x_2 & + & 8x_3 & = & 50 & \text{(IV)}\\
 & & & & & & x_1, x_3 & \geq & 0 & \text{(V)}\\
 & & & & & & x_2 & & \text{urs} & \text{(VI)}
\end{array}$$

In **Step 2** we have to remove double inequalities. Constraint (III) is a double inequality, and so is also constraint (IV). Rewriting the double inequalities as two inequalities each, we obtain the following form or our LP:

$$\begin{array}{lrrrrrrrll}
\max\ -z = & & -11x_1 & + & 10x_2 & - & 7x_3 & & & \text{(I')}\\
\text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 & \text{(II)}\\
 & & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III.1')}\\
 & 2 \leq & x_1 & & & + & 6x_3 & & & \text{(III.2')}\\
 & & x_1 & - & 4x_2 & + & 8x_3 & \leq & 50 & \text{(IV.1')}\\
 & 50 \leq & x_1 & - & 4x_2 & + & 8x_3 & & & \text{(IV.2')}\\
 & & & & & & x_1, x_3 & \geq & 0 & \text{(V)}\\
 & & & & & & x_2 & & \text{urs} & \text{(VI)}
\end{array}$$

Putting the inequalities (upper and lower bounds) so that the variables are all on the right-hand side, we obtain

$$\begin{array}{lrrrrrrrl}
\max\ -z = & -11x_1 & + & 10x_2 & - & 7x_3 & & & \text{(I')}\\
\text{s.t.} & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 & \text{(II)}\\
 & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III.1')}\\
 & x_1 & & & + & 6x_3 & \geq & 2 & \text{(III.2'')}\\
 & x_1 & - & 4x_2 & + & 8x_3 & \leq & 50 & \text{(IV.1')}\\
 & x_1 & - & 4x_2 & + & 8x_3 & \geq & 50 & \text{(IV.2'')}\\
 & & & & & x_1, x_3 & \geq & 0 & \text{(V)}\\
 & & & & & x_2 & & \text{urs} & \text{(VI)}
\end{array}$$

Step 2 is now finished.

In **Step 3** we have to remove lower bounds ($\geq$) by multiplying inequalities by -1 when needed. We need to do this for the inequalities (II), (III.2") and (IV.2"). We obtain the LP

$$
\begin{array}{llrcrcrcrl}
\max & -z = & -11x_1 & + & 10x_2 & - & 7x_3 & & & \text{(I')} \\
\text{s.t.} & & -9x_1 & - & 5x_2 & + & 3x_3 & \leq & -21 & \text{(II')} \\
 & & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III.1')} \\
 & & -x_1 & & & - & 6x_3 & \leq & -2 & \text{(III.2"')} \\
 & & x_1 & - & 4x_2 & + & 8x_3 & \leq & 50 & \text{(IV.1')} \\
 & & -x_1 & + & 4x_2 & - & 8x_3 & \leq & -50 & \text{(IV.2"')} \\
 & & & & & & x_1, x_3 & \geq & 0 & \text{(V)} \\
 & & & & & & x_2 & & \text{urs} & \text{(VI)}
\end{array}
$$

Finally, in **Step 4** we need to impose sign constraints to all the decision variables. For x_1 and x_3 we already have sing constraints, but x_2 is unrestricted in sign (urs). Therefore, to get rid of the "anti-constraint" (VI) we need to replace $x_2 = x_2^+ - x_2^-$ everywhere and add the sign constrains $x_2^+, x_2^- \geq 0$. To make completely clear what happens, we do this in two substeps. First substep is almost a "search and replace". We obtain

$$
\begin{array}{llrcrcrcrl}
\max & -z = & -11x_1 & + & 10(x_2^+ - x_2^-) & - & 7x_3 & & & \text{(I')} \\
\text{s.t.} & & -9x_1 & - & 5(x_2^+ - x_2^-) & + & 3x_3 & \leq & -21 & \text{(II')} \\
 & & x_1 & & & + & 6x_3 & \leq & 40 & \text{(III.1')} \\
 & & -x_1 & & & - & 6x_3 & \leq & -2 & \text{(III.2"')} \\
 & & x_1 & - & 4(x_2^+ - x_2^-) & + & 8x_3 & \leq & 50 & \text{(IV.1')} \\
 & & -x_1 & + & 4(x_2^+ - x_2^-) & - & 8x_3 & \leq & -50 & \text{(IV.2"')} \\
 & & & & & & x_1, x_2^+, x_2^-, x_3 & \geq & 0 & \text{(V')}
\end{array}
$$

In the second substep we "open the parentheses". This is the final step, and we obtain the standard form LP

$$
\begin{array}{llrcrcrcrcrl}
\max & -z = & -11x_1 & + & 10x_2^+ & - & 10x_2^- & - & 7x_3 & & & \text{(I")} \\
\text{s.t.} & & -9x_1 & - & 5x_2^+ & + & 5x_2^- & + & 3x_3 & \leq & -21 & \text{(II")} \\
 & & x_1 & & & & & + & 6x_3 & \leq & 40 & \text{(III.1")} \\
 & & -x_1 & & & & & - & 6x_3 & \leq & -2 & \text{(III.2"")} \\
 & & x_1 & - & 4x_2^+ & + & 4x_2^- & + & 8x_3 & \leq & 50 & \text{(IV.1")} \\
 & & -x_1 & + & 4x_2^+ & - & 4x_2^- & - & 8x_3 & \leq & -50 & \text{(IV.2"")} \\
 & & & & & & & & x_1, x_2^+, x_2^-, x_3 & \geq & 0 & \text{(V')}
\end{array}
\tag{8.12}
$$

Thus we have found the standard form (8.12) for the LP (8.7).

An LP is in **dual standard form** if it is of the form

$$
\begin{array}{lrcl}
\min & z = \mathbf{c}'\mathbf{x} & & \\
\text{s.t.} & \mathbf{Ax} & \geq & \mathbf{b} \\
 & \mathbf{x} & \geq & \mathbf{0}
\end{array} .
\tag{8.13}
$$

Any LP can be transformed into a dual standard form. Indeed, it is not difficult to see how to modify Algorithm 8.10 so that it gives the dual standard form.

8.14 Remark (Primal Standard Form)
To compare with the dual standard from (8.13), the standard form (8.1) is sometimes called the **primal standard form**.

Finally, let us consider the so-called **slack forms**, or **slack & surplus forms** for LP's. The reason behind these forms is that "inequalities are nasty – equalities are nice", especially for computers.

The idea of the slack forms is that by using auxiliary decision variables s_i^+ for the constraints i, we can write any upper bound inequality constraint

$$A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n \ \leq \ b_i$$

as an equality constraint

$$A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n + s_i^+ \ = \ b_i,$$

where $s_i^+ \geq 0$ is the **slack** in the constraint. Therefore s_i^+ is a non-negative dummy decision variable that we can choose. The inequality constraint becomes an equality constraint where we can choose how much slack there is below the upper limit b_i.

In the same way, any lower bound inequality

$$A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n \ \geq \ b_i$$

can be written as an equality

$$A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}x_n - s_i^- \ = \ b_i,$$

where $s_i^- \geq 0$ is the **surplus**, or **excess** in the constraint. Thus s_i^- is a non-negative dummy decision variable we can choose. This means we can choose how much will the bound b_i will be exceeded.

An LP is in a **slack form** if all of its constraints are equalities and all the decision variables are restricted in sign to be non-negative. So, we only insist equality constraints and sign constraints. In particular, a slack form can be either maximization or minimization. **Any LP can be written in a slack form**. Indeed, this can be done by introducing for each inequality constraint i a unique slack or surplus variable, s_i^+ or s_i^-, as explained above. We do not give an explicit algorithm here. We hope that the next example is enough to make the transformation clear.

8.15 Example (Slack Form Transformation)
Consider the LP

$$\begin{array}{lrrrrrrrl}
\min z & = & 8x_1 & - & 8x_2 & + & 7x_3 & & \\
\text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3 & \geq & 14 \\
 & & x_1 & & & + & 6x_3 & \leq & 42 \ . \\
 & & & & & & x_1, x_2 & \geq & 0 \\
 & & & & & & x_3 & & \text{urs}
\end{array}$$

We want to write this LP in a slack form. We need to impose the sign constraint to x_3 and to introduce surplus to the first constraint, and slack to the second constraint. Let us first impose the sign constraint by splitting x_3 as $x_3 = x_3^+ - x_3^-$. We obtain the LP

$$\begin{array}{lrcrcrcrcr} \min z & = & 8x_1 & - & 8x_2 & + & 7x_3^+ & - & 7x_3^- & & \\ \text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3^- & + & 3x_3^- & \geq & 14 \\ & & x_1 & & & + & 6x_3^+ & - & 6x_3^- & \leq & 42 \\ & & & & & & & & x_1, x_2, x_3^-, x_3^+ & \geq & 0 \end{array} .$$

Then we add surplus s_1^- to the first constraint and slack s_2^+ to the second constraint. We obtain the slack form LP

$$\begin{array}{lrcrcrcrcrcrcr} \min z & = & 8x_1 & - & 8x_2 & + & 7x_3^+ & - & 7x_3^- & & & & & & \\ \text{s.t.} & & 9x_1 & + & 5x_2 & - & 3x_3^- & + & 3x_3^- & - & s_1^- & & & = & 14 \\ & & x_1 & & & + & 6x_3^+ & - & 6x_3^- & & & + & s_2^+ & = & 42 \\ & & & & & & & & & & & & x_1, x_2, x_3^-, x_3^+, s_1^-, s_2^+ & \geq & 0 \end{array} .$$

8.16 Remark (Pure Slack Form)
It is possible to write any LP in the so-called **pure slack form** with only slack variables $s_i = s_i^+$ without the surplus variables s_i^-. Indeed, if we first transform the LP into a primal standard form, then we have only upper bounds, and consequently no need for the surplus variables. By using block matrix formalism this means that the primal standard form LP

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

can be written a the pure slack form as

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

where $\mathbf{I}$ is the identity matrix.

8.17 Exercise (Primal Standard Form, Dual Standard Form, and Slack Form)
Consider the LP

$$\begin{array}{lrcrcrcrcr} \max z & = & & & & 10x_2 & + & 7x_3 & & \\ \text{s.t.} & & & 4x_1 & + & 5x_2 & - & 3x_3 & \geq & 21 \\ & 2 & \leq & x_1 & & & + & 6x_3 & \leq & 40 \\ & & & x_1 & - & 4x_2 & + & 8x_3 & = & 50 \\ & & & & & & & x_1, x_3 & \geq & 0 \\ & & & & & & & x_2 & & \text{urs} \end{array}.$$

Express this LP in (i) primal standard form, (ii) dual standard form, and (iii) slack form. Solve all these three different (but equivalent) forms with glpk.

8.18 Exercise (GLPK Standard Form)
If glpk is called with only three input arguments — **c**, **A**, and **b** — then it assumes that the LP is in what it calls a "standard LP":

$$\begin{array}{lrcl} \min & z = \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}. \tag{8.19}$$

Transform the LP (8.7) of Problem 8.6 into this GLPK standard form and solve it by using glpk in the form

```
[x_min, z_min] = glpk(c, A, b)
```

Duality Theorems

The **dual** of an LP is another LP that is constructed from the original (the **primal**) LP according to the following rules:

- The objective is inversed: maximum in the primal LP becomes minimum in the dual LP, and vice versa.
- Decision variables in the primal LP become constraints in the dual LP.
- Constraints in the primal LP become decision variables in the dual LP.
- Upper bounds are turned into lower bounds, and vice versa.
- Technology matrix is transposed.
- Sign constraints are kept.

More precisely, if the **primal** LP is given in the primal standard form

$$\begin{array}{lrcl} \max & z = \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \le & \mathbf{b} \\ & \mathbf{x} & \ge & \mathbf{0} \end{array}, \tag{8.20}$$

then its **dual** LP is in the dual standard from

$$\begin{array}{lrcl} \min & w = \mathbf{b}'\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}'\mathbf{y} & \ge & \mathbf{c} \\ & \mathbf{y} & \ge & \mathbf{0} \end{array}. \tag{8.21}$$

8.22 Example (Dual Transformation)
Consider the LP

$$\begin{array}{llrcrcrcr} \min & z = & 50x_1 & + & 20x_2 & + & 30x_3 & & \\ \text{s.t.} & & 2x_1 & + & 3x_2 & + & 4x_3 & \ge & 11 \\ & & 12x_1 & + & 13x_2 & + & 14x_3 & \le & 111 \\ & & x_1 & + & x_2 & + & x_3 & = & 1 \\ & & & & & & x_1, x_2, x_3 & \ge & 0 \end{array}$$

The LP 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 right now. So, here is the standard form:

$$\begin{array}{llrcrcrcr} \max & -z = & -50x_1 & - & 20x_2 & - & 30x_3 & & \\ \text{s.t.} & & -2x_1 & - & 3x_2 & - & 4x_3 & \le & -11 \\ & & 12x_1 & + & 13x_2 & + & 14x_3 & \le & 111 \\ & & x_1 & + & x_2 & + & x_3 & \le & 1 \\ & & -x_1 & - & x_2 & - & x_3 & \le & -1 \\ & & & & & & x_1, x_2, x_3 & \ge & 0 \end{array}$$

Now we are ready to present the dual:

$$\begin{array}{llrcrcrcrcr} \min & -w = & -11y_1 & + & 111y_2 & + & y_3 & - & y_4 & & \\ \text{s.t.} & & -2y_1 & + & 12y_2 & + & y_3 & - & y_4 & \ge & -50 \\ & & -3y_1 & + & 13y_2 & + & y_3 & - & y_4 & \ge & -20 \\ & & -4y_1 & + & 14y_2 & + & y_3 & - & y_4 & \ge & -30 \\ & & & & & & & & y_1, y_2, y_3, y_4 & \ge & 0 \end{array}.$$

(We used variable $-w$ in the dual because there was variable $-z$ in the standard form primal). Note now that the dual LP is in dual standard form: It is a minimization problem with only inequalities of type $\geq$. The original primal LP was also 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 above into a maximization problem with $\leq$ type inequalities. In fact, let us transform the dual LP into a standard form. We obtain

$$\begin{array}{llrcrcrcrcl} \max & w = & 11y_1 & - & 111y_2 & - & y_3 & + & y_4 & & \\ \text{s.t.} & & 2y_1 & - & 12y_2 & - & y_3 & + & y_4 & \leq & 50 \\ & & 3y_1 & - & 13y_2 & - & y_3 & + & y_4 & \leq & 20 \; . \\ & & 4y_1 & - & 14y_2 & - & y_3 & + & y_4 & \leq & 30 \\ & & & & & & & & y_1, y_2, y_3, y_4 & \geq & 0 \end{array}$$

8.23 Remark (Dual is Not Primal)
Note that the dual LP (8.21) is **not** the same problem as the primal LP (8.20). So, the LP (8.21) is **not** the LP (8.20) written in a different, but equivalent, form. Nevertheless the primal and the dual are very closely related, as explained in Theorem 8.27 later in this section.

8.24 Remark (Dual of Dual is Primal)
The duality transformation (8.20)$\rightsquigarrow$(8.21) is convenient in the sense that its inverse is twice itself. Indeed, suppose we have started with the LP (8.20) and we have now the LP (8.21). In order to perform the dual transformation to the LP (8.21) we can first rewrite it in the (primal) standard form by using Algorithm 8.10. So, we have the (dual) LP

$$\begin{array}{lrcl} \min & w = \mathbf{b}'\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}'\mathbf{y} & \geq & \mathbf{c} \; . \\ & \mathbf{y} & \geq & \mathbf{0} \end{array}$$

By using Step 1 of Algorithm 8.10 we can write this LP equivalently as

$$\begin{array}{lrcl} \max & -w = -\mathbf{b}'\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}'\mathbf{y} & \geq & \mathbf{c} \; . \\ & \mathbf{y} & \geq & \mathbf{0} \end{array}$$

Step 2 of Algorithm 8.10 is not needed. Step 3 gives us

$$\begin{array}{lrcl} \max & -w = -\mathbf{b}'\mathbf{y} & & \\ \text{s.t.} & -\mathbf{A}'\mathbf{y} & \leq & -\mathbf{c} \; . \\ & \mathbf{y} & \geq & \mathbf{0} \end{array}$$

Step 4 of Algorithm 8.10 is not needed. Indeed, we have a (primal) standard from. Now, we can formally perform the dual transformation to this standard form. Since $\mathbf{A}'' = \mathbf{A}$, we obtain, by switching the roles of $\mathbf{x}$ and $\mathbf{y}$, and w and z, that

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

Now we should rewrite this LP in the (primal) standard from. This can be done by using Algorithm 8.10 again. Only Step 1 and Step 3 are needed. We obtain

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

So, we are back to the LP (8.20) where we started with.

8.25 Exercise (Find the Dual)
Find the dual LP of the following LP:

$$\begin{array}{lccccccccc} \min z & = & -5x_1 & +3x_2 & & -x_4 & & +3x_6 & & & \\ \text{s.t.} & & & -x_2 & +7x_3 & & -2x_5 & & -2x_7 & \geq & 13 \\ & & x_1 & & -2x_3 & +4x_4 & & & +7x_7 & \leq & 74 \\ & & & & & & & & x_1,x_2,x_3,x_4,x_5,x_6,x_7 & \geq & 0 \end{array}. \tag{8.26}$$

Note that it is extremely helpful, but not strictly necessary, to rewrite the LP (8.26) first in the primal standard form.

In the following theorem we see how the primal LP and the dual LP solutions go nicely hand-in-hand.

8.27 Theorem (Duality Theorem)

$$z^*_{\text{primal}} = w^*_{\text{dual}} \tag{8.28}$$
$$\mathbf{x}^*_{\text{primal}} = \boldsymbol{\lambda}_{\text{dual}} \tag{8.29}$$
$$\boldsymbol{\lambda}_{\text{primal}} = \mathbf{y}^*_{\text{dual}} \tag{8.30}$$
$$\mathbf{u}_{\text{primal}} = \mathbf{s}^{-*}_{\text{dual}} \tag{8.31}$$
$$\mathbf{s}^{+*}_{\text{primal}} = \mathbf{u}_{\text{dual}}. \tag{8.32}$$

Every theorem needs a formal and rigorous proof. The proof of Theorem 8.27 is left for the students who want to complete the course without the weekly exercises. In these notes we just ask a reasonable argument for why Theorem 8.27 is true by using Problem 5.1 and it dual, Problem 6.6, as an exercise.

8.33 Exercise (Muad'Dib Bakery–Sietch Tabr Duality)
Consider Problem 5.1 (Muad'Dib Bakery) and Problem 6.6 (Sietch Tabr Nutrient Diet). These problems are in duality. Check that Duality Theorem 8.27 holds true with Problem 5.1 as the primal LP and Problem 6.6 as the dual LP.

8.34 Remark (Unboundedness and Infeasibility in Duality)
An LP can be unbounded (meaning that the optimal solution for maximization is, in principle, $+\infty$ and for minimization, in principle, $-\infty$) or infeasible (meaning there are no solutions at all). The duality theorem tells us that

- If the primal LP is unbounded, then the dual LP is infeasible.
- If the dual LP is unbounded, then the primal LP is infeasible.

You should be careful about the logic here! It is possible for both the dual LP and the primal LP to be infeasible at the same time.

8.35 Exercise (Dual Jihad)
We have learned that every LP has a dual LP. Construct the dual LP of Muad'Dib's Jihad LP of Problem 7.1. Solve this dual LP, and compare the solution with Solution 7.4. Try to interpret the dual LP somehow, if possible.

Chapter 9

Transportation Problems

"If the only tool you have is a hammer, you will start treating all your problems like a nail." Fair enough, but the LP way of thinking is a pretty good hammer. In this chapter we will treat transportation problems as LP's. There are also specialized algorithms for solving these problems that work much faster than the general LP algorithms, but of course they work only in the special type of problems.

In a transportation problem one has to ship products from ports (or sources) to markets (or destinations) so that the total shipping cost is as small as possible while still meeting the demands of each market (destination). The basic ingredients (the data) of a transportation problem are

- **Supplies** of the sources (or ports).
- **Demands** of the destinations (or markets).
- **Shipping costs** (or route costs) per unit from sources to destinations.

More formally, in a transportation problem we have

- n sources $(i = 1, \ldots, n)$ each having supply denoted by s_i. Thus the supplies are given by a vector $\mathbf{s}$.
- m destinations $(j = 1, \ldots, m)$ each having demand denoted by d_j. Thus the demands are given by a vector $\mathbf{d}$.
- Shipping costs per unit $(i = 1, \ldots, n,\ j = 1, \ldots, m)$ from the source i to the destination j denoted by C_{ij}. Thus the shipping costs are given by a matrix $\mathbf{C}$.

To solve a transportation problem as an LP we must first express it in a LP form. In practice this means figuring out the typical LP parameters $\mathbf{c}, \mathbf{A}, \mathbf{b}$ from the transportation parameters $\mathbf{s}, \mathbf{d}, \mathbf{C}$. Finding $\mathbf{c}$ and $\mathbf{b}$ are not too difficult. Finding $\mathbf{A}$ is, unfortunately, rather complicated.

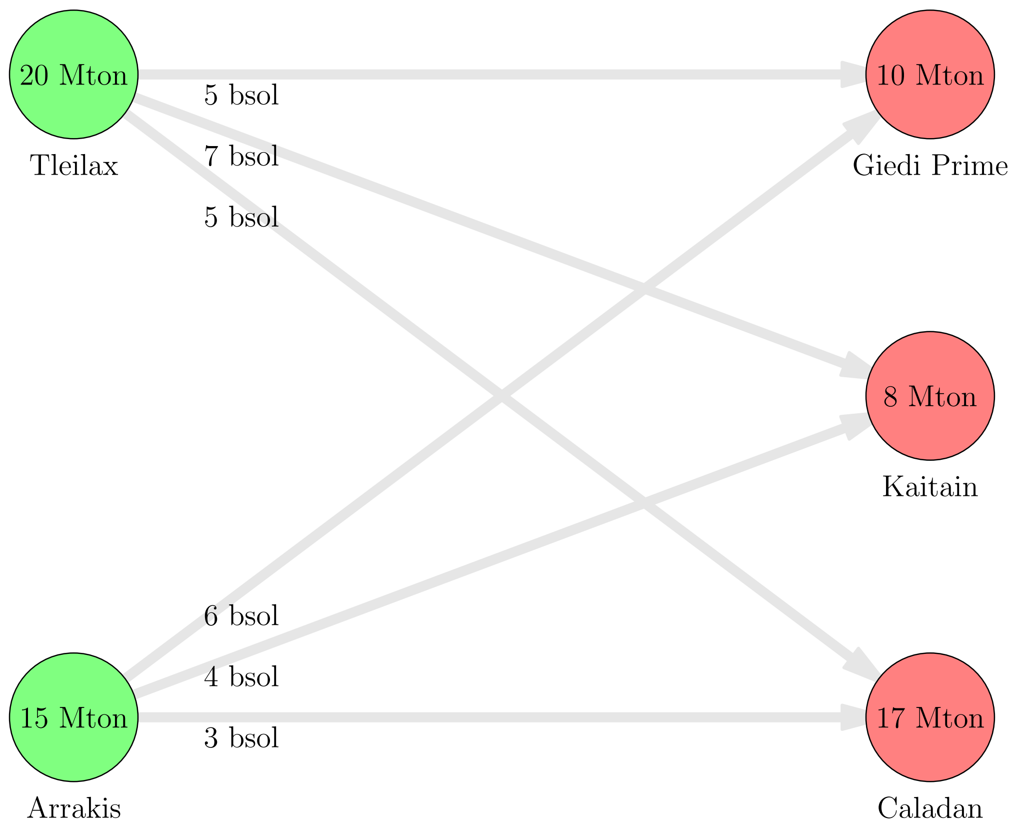
Primal Transportation Problem

We start with a problem description that is given in a pretty annoying form. (It was probably written by the Imperial Legal Department.)

9.1 Problem (Spice Must Flow with Heighliners)
The Spacing Guild transports Spice Melange from Arrakis and Tleilax with its heighliners. The spice is transported to Caladan, Kaitain, and Giedi Prime. In Arrakis there are 15 megatons (15 Mton) of spice to be transported. In Tleilax there are 20 megatons (20 Mton) to be transported. The Spacing Guild has promised to transport 17 megatons (17 Mton) to Caladan, 8 megatons (8 Mton) to Kaitain, and 10 megatons (10 Mton) to Giedi Prime. The shipping costs are 3 billion solaris (3 bsol) per megaton from Arrakis to Caladan, 4 billion solaris (4 bsol) per megaton from Arrakis to Kaitain, and 6 billion solaris (6 bsol) per megaton from Arrakis to Giedi Prime. The shipping costs from Tleilax are 5 billion solaris (6 bsol) per megaton to Caladan, 7 billion solaris (7 bsol) per megaton to Kaitain, and 5 billion solaris (5 bsol) per megaton to Giedi Prime. The Spacing guild want to minimize its transportation costs. How should the Spacing Guild transport the spice from Tleilax and Arrakis to Giedi Prime, Kaitain and Caladan?

Problem 9.1 above was given in an annoying form in purpose. It had a lot of stuff that was simply not relevant. Also, the data was hidden inside the non-essential babbling.

One natural way to represent the data of Problem 9.1 is to realize that we are working with a flow network. Thus, the following representation is natural:



Another natural way of representing the data in Problem 9.1 is the following tabular form:

Heighliner Spice Transportation				
Source/Destination	Caladan	Kaitain	Giedi Prime	**Supply**
Arrakis	3 bsol	4 bsol	6 bsol	15 Mton
Tleilax	5 bsol	7 bsol	5 bsol	20 Mton
Demand	17 Mton	8 Mton	10 Mton	

Problem 9.1 (Spicy Must Flow with Heighliners) is a typical **transportation problem**, and it, as all transportation problems, can be modeled as an LP. Let us follow Algorithm 5.2 to see how to do this:

We start with a clever trick. We set

$$X_{ij} \quad = \quad \text{Mtons of Spice shipped from source } i \text{ to destination } j.$$

These are the **decision variables** for Problem 9.1, or for any transportation problem for that matter, save the actual product to be shipped. We can enumerate the sources and destinations any way we want. We choose to use the numeration corresponding to the tabular form, or read the graph from the bottom up. Thus, we have $2 \times 3 = 6$ decision variables:

$$\begin{aligned}
X_{11} &= \text{Mtons of Spice shipped from Arrakis to Caladan},\\
X_{12} &= \text{Mtons of Spice shipped from Arrakis to Kaitain},\\
X_{13} &= \text{Mtons of Spice shipped from Arrakis to Giedi Prime},\\
\\
X_{21} &= \text{Mtons of Spice shipped from Tleilax to Caladan},\\
X_{22} &= \text{Mtons of Spice shipped from Tleilax to Kaitain},\\
X_{23} &= \text{Mtons of Spice shipped from Tleilax to Giedi Prime}.
\end{aligned}$$

9.2 Remark (Cleverness Comes with a Price)
Note the clever use of indexes here: we consider the decision variables in a matrix form. There is a small price we have to pay for this cleverness later, however.

The **objective** of any transportation problem is to minimize the total shipping cost. So, the objective is (in general form)

$$\min z \quad = \quad \sum_i \sum_j C_{ij} X_{ij}$$

where

$$C_{ij} \quad = \quad \text{the cost of shipping one Mton of Spice from source } i \text{ to destination } j.$$

Since

$$\mathbf{C} \quad = \quad \begin{bmatrix} 3 & 4 & 6 \\ 5 & 7 & 5 \end{bmatrix},$$

the objective for the particular case of Problem 9.1 is

$$\min z \;=\; 3X_{11}+4X_{12}+6X_{13}+5X_{21}+7X_{22}+5X_{23}.$$

What about the **constraints** then?

There are the supply and demand constraints.

Each source has only so many Mtons of Spice it can supply. So, if s_i is the supply limit for source i then we have the **supply constraints** (in general form)

$$\sum_j X_{ij} \;\le\; s_i \qquad \text{for all sources } i.$$

(Please, do not confuse s_i with a slack variable here. We are sorry for the clashing notation!) Since there are two sources, we have two supply constraints. Plugging in the numbers, we obtain

$$\begin{aligned} X_{11}+X_{12}+X_{13} &\le 15, \\ X_{21}+X_{22}+X_{23} &\le 20. \end{aligned}$$

Each destination also demands so many Mtons of Spice, and according to the problem we are committed to meet these demands. So, if d_j is the demand for Spice in the destination j then we have the **demand constraints** (in general form)

$$\sum_i X_{ij} \;\ge\; d_j \qquad \text{for all destinations } j.$$

In Problem 9.1 we have 3 destinations with demands 17 Mton, 8 Mton and 10 Mton. Thus, we have the particular demand constraints

$$\begin{aligned} X_{11}+X_{21} &\ge 17, \\ X_{12}+X_{22} &\ge 8, \\ X_{13}+X_{23} &\ge 10. \end{aligned}$$

Finally, there are of course the **sign constraints** (in general form)

$$X_{ij} \;\ge\; 0 \qquad \text{for all sources } i \text{ and destinations } j.$$

Indeed, it would be pretty hard to transport negative amounts of Spice. So, in the particular case of Problem 9.1 we have the six sign constraints

$$X_{11}, X_{12}, X_{13}, X_{21}, X_{22}, X_{23} \;\ge\; 0.$$

Finally, here is the LP model for Problem 9.1:

$$\begin{array}{lrrrrrrrr}
\min z = & 3X_{11} & +4X_{12} & +6X_{13} & +5X_{21} & +7X_{22} & +5X_{23} & & \\
\text{s.t.} & X_{11} & +X_{12} & +X_{13} & & & & \le & 15 \\
 & & & & X_{21} & +X_{22} & +X_{23} & \le & 20 \\
 & X_{11} & & & +X_{21} & & & \ge & 17 \\
 & & X_{12} & & & +X_{22} & & \ge & 8 \\
 & & & X_{13} & & & +X_{23} & \ge & 10 \\
 & X_{11}, & X_{12}, & X_{13}, & X_{21}, & X_{22}, & x_{23} & \ge & 0
\end{array} \quad . \tag{9.3}$$

Note how we have used columns to put the decision variables X_{ij} in their right place by "unfolding" the matrix $\mathbf{X} = [X_{ij}]_{i=1,2;j=1,2,3}$ into a vector.

Next we solve the LP formulation (9.3) of Problem 9.1 by using glpk in a quick and dirty way.

9.4 Solution (Spice Must Flow with Heighliners, Quick and Dirty)
The following scrpit m-file heighliner_qd.m, which can be downloaded from http://lipas.uwasa.fi/~tsottine/spicy_or/heighliner_qd.m solves Problem 9.1 in a quick and dirty way.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Spice must Flow with Heighliners, quick and dirty solution
%% (Problem 9.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s = [15 20]';                        %% Supplies in Mtons.
d = [17  8 10]';                     %% Demands in Mtons.
C = [                                %% Route costs bsol per Mton.
3   4   6;
5   7   5
];

c = C'(:);                           %% Unfold matrix C into a vector c.
A = [
1   1   1   0   0   0;               %% Supply constraint for source 1.
0   0   0   1   1   1;               %% Supply constraint for source 2.
1   0   0   1   0   0;               %% Demand constraint for destination 1.
0   1   0   0   1   0;               %% Demand constraint for destination 2.
0   0   1   0   0   1;               %% Demand constraint for destination 3.
];
b = [s;d];                           %% Upper and lower bounds.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve with glpk, reshape the solution and print out.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[X, cost] = glpk(c, A, b, [], [], "UULLL", "CCCCCC");
X = reshape(X,3,2)'                  %% Reshape and print out.
cost                                 %% Print out.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test result:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%>> heighliner_qd
%%X =
%%
%%     7     8     0
%%    10     0    10
%%
```

```
46 %%cost = 153
```

Let us explain this quick and dirty solution a little bit. Lines 1–10 are comments for the human reader. Nothing new there. Lines 12–17 set the transportation problem data. This should be clear also. The line 19 is something new. This is where we pay the price of having our transportation costs in a matrix form. What happens in the line 19 is that we unfold the matrix `C` into a vector by glueing its rows one after each other. This unfolding would happen with the construct `C(:)`. Unfortunately this works the "wrong way around": `C(:)` would work comlumn-wise and we want it to work row-wise. The obvious solution is to transpose the original matrix and the apply the unfolding. This is precisely what line the 19 does. The most complicated and tedious part of the script is the lines 22–26 where we build the technology matrix `A`. The first 2 rows of matrix `A` correspond to the supply side constraints. Compare this with the LP (9.3) and the "$\le$" constraints there. Similarly, the last 3 rows of the matrix `A` correspond to the demand side constraints. Compare this with the LP (9.3) and the "$\ge$" constraints there. The LP bounds `b` should be obvious from the LP formulation (9.3): first we have the supply constraints `s` and then we have the demand constraints `d`. The beef-line 33 should be clear by now. Finally, in the line 34 we reshape the vector `X` into a matrix `X`. This is the price we have to pay for our cleverness of having the decision variables as a matrix. The reader is strongly recommended to type **help reshape** on GUN Octave command prompt and run some examples to understand what this reshaping means.

Finally, let consider the **solution**. The minimal shipping cost is 153 bsol and the optimal shipping schedule is

$$\mathbf{X} = \begin{bmatrix} 7 & 8 & 0 \\ 10 & 0 & 10 \end{bmatrix}$$

meaning that, for example, $X_{11} = 7$ is the Mtons of Spice transported from Arrakis to Caladan. To make the meaning of $\mathbf{X}$ completely clear, let us give the optimal transportation schedule in a tabular form with the names attached:

Heighliner Spice Transportation (Optimal schedule)				
Source/Destination	Caladan	Kaitain	Giedi Prime	**Supply**
Arrakis	3 bsol (7 Mton)	4 bsol (8 Mton)	6 bsol (0 Mton)	15 Mton
Tleilax	5 bsol (10 Mton)	7 bsol (0 Mton)	5 bsol (10 Mton)	20 Mton
Demand	17 Mton	8 Mton	10 Mton	

9.5 Exercise (Excess Spice)

Problem 9.1 was **balanced**: the total spice supply was the same as the total spice demand. Suppose now that there are 22 Mton of spice in Arrakis. So, there is 7 Mton excess supply. What is the optimal transportation schedule now? What happens to the excess spice?

9.6 Exercise (Arrakis–Kaitain Blockade)

Consider Problem 9.1 of transporting spice. Suppose the route from Arrakis to Kaitain is blocked. What is the new optimal transportation schedule?

Hint: Make the transportation cost from Arrakis to Kaitain so big that it will not be used in the optimal solution. This is what is typically called the **big-M trick**.

Finally, let us criticize the quick and dirty solution 9.4. What was so quick and dirty there? Obviously the fact that the solution was ad hoc. It only works for the special problem. It would be nice to have a function, let us call it transsolver, that takes the transportation data $\mathbf{s}, \mathbf{d}, \mathbf{C}$ as its arguments and returns the optimal schedule and its associated cost.

Here is the function transsolver. You can, and indeed should, download it from http://lipas.uwasa.fi/~tsottine/spicy_or/transsolver.m. You can also use it to solve the exercises.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Function [schedule, cost] = transsolver(s, d, C) solves the transportation
%% problem
%%
%%        _____   _____
%%          \       \
%%    min    >       >     C(i,j)*X(i,j)
%%          /       /
%%         _____   _____
%%            i       j
%%
%%           _____
%%            \
%%    s.t.     >    X(i,j)  <=   s(i)  for  all  ports  i
%%            /
%%           _____
%%              j
%%
%%           _____
%%            \
%%             >    X(i,j)  >=   d(j)  for  all  markets  j
%%            /
%%           _____
%%              i
%%
%% Input:
%%
%% s, d, C
%%      Supply vector, demand vector, and cost matrix.
%%
%% Output:
%%
%% schedule, cost
%%      The optimal schedule and the associated minimal transportation cost.
%%
%% See also:  glpk.
function [schedule, cost] = transsolver(s, d, C)
    %% Some short-hands.
```

```
39     m1 = length(s);                                     %% Number of sources.
40     m2 = length(d);                                     %% Number of destinations.
41     n = m1*m2;                                          %% Number of decisions.
42 
43     %% Build the technology (flow-constraint) matrix A
44     A = zeros(m1+m2,n);                                 %% Initialization.
45     for i = 1:m1
46         A(i, (1:m2)+(i-1)*m2) = ones(1,m2);             %% Supply side.
47         A((1:m2)+m1, (1:m2)+(i-1)*m2) = eye(m2);        %% Demand side.
48     end
49 
50     %% Set the sense of the bounds (upper for sources, lower for destinations).
51     ctype = "";
52     for i=1:m1
53         ctype = [ctype "U"];
54     end
55     for i=1:m2
56         ctype = [ctype "L"];
57     end
58 
59     %% Set the type of each decision variable as continuous.
60     vtype = "";
61     for i=1:n
62         vtype = [vtype "C"];
63     end
64 
65     %% Solve the system by calling glpk.
66     [schedule, cost] = glpk(C'(:), A, [s; d], [], [], ctype, vtype);
67 
68     %% Reshape the schedule vector to a matrix.
69     schedule = reshape(schedule, m2, m1)';
70 end
```

The workings of transsolver should be relatively obvious, except maybe for the lines 44–48, where the technology matrix A is constructed. To understand what is going on there, the reader is invited to compare this with the quick and dirty solution 9.4 and try to see how this generalizes what was going on there.

Let us then solve Problem 9.1 by using transsolver to see how convenient it is.

9.7 Solution (Spice Must Flow with Heighliners)

Here is the script m-file heighliner.m that solves Problem 9.1 by using our hand-made function transsolver. You can download the script file from http://lipas.uwasa.fi/~tsottine/spicy_or/heighliner.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Spice must Flow with Heighliners, solution with transsolver.
%% (Problem 9.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
s = [15 20]';                    %% Supplies in Mtons.
d = [17  8 10]';                 %% Demands in Mtons.
C = [                            %% Route costs bsol per Mton.
3  4  6;
5  7  5
];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve with transsolver and print out the solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[X, cost] = transsolver(s,d,C)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test result:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%>> heighliner
%%X =
%%
%%     7     8     0
%%    10     0    10
%%
%%cost = 153
```

Dual Transportation Problem

Every LP has a dual LP, this someting we know. Transportation problem can be modeled as an LP. This is also something we know. That is actually why we called (in the section title) the transportation problem as the “primal” transportation problem. But what is exactly a dual transportation problem?

Recall that in the dual problem constraints become variables and variables be come constraints. Also the sense, maximization or minimization, is reversed. Since the primal transportation problem was a minimization, we expect the dual transportation problem to be a maximization. Also, the constraints in the primal were twofold: supplies and demands. So, we expect to have in the dual two kinds of decision variables: those related to supplies and those related to demands. Let us denote by $\mathbf{u}$ (sorry for the clashing notation with reduced costs!) the dual decision variables associated with the supply constraints, and let us denote by $\mathbf{v}$ the dual decision variables associated with the demand constraints. The constraints in the dual transportation problem will be associated with the objective function of the primal transportation problem. So, we expect to have constraints for all C_{ij} for different i and j.

Let us then try to guess what the dual transportation problem is. Let us first write the primal transportation problem in a form that resembles a dual standard form (since it is a minimization problem). The normal LP formulation for the primal transportation problem is

$$\begin{array}{lrcll} \min z = \sum_i \sum_j C_{ij}X_{ij} & & & & \\ \text{s.t.} & \sum_j X_{ij} & \le & s_i & \text{for all } i, \\ & \sum_i X_{ij} & \ge & d_j & \text{for all } j, \\ & X_{ij} & \ge & 0 & \text{for all } i \text{ and } j. \end{array}$$

This is pretty close to a dual standard form. Indeed, if we do not mind the matrix formulation, we can rewrite the primal transportation problem as

$$\begin{array}{lrcll} \min z = \sum_i \sum_j C_{ij}X_{ij} & & & & \\ \text{s.t.} & -\sum_j X_{ij} & \ge & -s_i & \text{for all } i, \\ & \sum_i X_{ij} & \ge & d_j & \text{for all } j, \\ & X_{ij} & \ge & 0 & \text{for all } i \text{ and } j. \end{array}$$

This is a dual standard from (although it might not be completely obvious). Therefore its dual (remember that the dual of a dual is primal) takes the following primal standard form

$$\begin{array}{lrcll} \max w = -\sum_i u_i s_i + \sum_j v_j d_j & & & & \\ \text{s.t.} & v_j - u_i & \le & C_{ij} & \text{for all } i \text{ and } j, \\ & u_i, v_j & \ge & 0 & \text{for all } i \text{ and } j. \end{array} \tag{9.8}$$

Now we know, formally, what the dual transportation problem is. It is the LP (9.8) above. But what does it mean? The next problem gives us one interpretation.

9.9 Problem (Spice Must Flow with No-Ships)
The Ixians have invented the no-ships that do not need the Spacing Guild Navigators. Thus they can compete in providing interstellar travel. The Ixians want to make an offer to take care of the spice flow of Problem 9.1. They offer to buy from the sources with unit prices u_i and sell in the destinations with unit prices v_j. What should be the actual offer so that the sources and destinations will abandon the Spacing Guild and turn their transportation over to the Ixians.

9.10 Remark (Heighliner–No-Ship Duality)
To understand the different points of view in Problem 9.1 and Problem 9.9 think that the Spacing Guild transportation costs are public information while the Ixian transportation costs are secret. The sources and destinations have already agreed on transporting the spice and are now only interested in the cost of transportation. They do not care how the transportation is handled.

Problem 9.9 is the dual of Problem 9.1. Indeed, the Ixians want to take over the business of transporting spice. If u_i is the offer per unit to buy from the source i and v_j is the offer per unit to sell for the destination j, then the total revenue the Ixians want to maximize is

$$\max w = -\sum_i u_i s_i + \sum_j v_j d_j.$$

Now, any source i can use the Spacing Guild to transport its spice to any destination j. The Spacing Guild price is C_{ij} per Mton of spice. Now the source–destination pair can consider whether to use the Ixian no-ships or the Spacing Guild heighliners. If the Ixian offer is such that

$$v_j - u_i \quad \leq \quad C_{ij},$$

then moving spice with the Ixian no-ships is cheaper than using the Spacing Guild heighliners. Consequently the Ixian can take over the transportation business.

9.11 Solution (Spice Must Flow with No-Ships, Quick and Dirty)

The following script m-file heighliner_qd.m, which can be downloaded by using the link http://lipas.uwasa.fi/~tsottine/spicy_or/noship_qd.m, solves Problem 9.9 in a quick and dirty way.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Spice must Flow with No-Ships, quick and dirty solution
%% (Problem 9.9 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s = [15  20]';                   %% Supplies in Mtons.
d = [17   8  10]';               %% Demands in Mtons.
C = [                            %% Route costs bsol per Mton.
3   4   6;
5   7   5
];

c = [-s; d];                     %% Buy with -u, sell with v.
A = [
    -1    0    1    0    0;      %% v1-u1
    -1    0    0    1    0;      %% v2-u1
    -1    0    0    0    1;      %% v3-u1
     0   -1    1    0    0;      %% v1-u2
     0   -1    0    1    0;      %% v2-u2
     0   -1    0    0    1       %% v3-u2
];
%% u1   u2   v1   v2   v3

%% Dirty but clear.  Compare with the rows of A above.
b = [
C(1,1);
C(1,2);
C(1,3);
C(2,1);
C(2,2);
C(2,3)
];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve with glpk, split the solution, and print out.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[uv, cost] = glpk(c, A, b, [], [], "UUUUUU", "CCCCC", -1);
u = uv(1:2)
v = uv(3:5)
cost

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test result:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%>> noship_qd
%%u =
%%
%%    2
%%    0
%%
%%v =
%%
%%    5
%%    6
%%    5
%%
%% cost = 153
```

There is really nothing new in this code that we have not yet seen. So, further explanations are probably not needed. Also, it should be clear why this code is quick and dirty: it only works for this very specific case.

As for the **solution**, the prices per Mton of spice offered for the sources (to buy) and to the destinations (to sell) are

Arrakis Buy with 2 bsol per Mton.
Tleilax Transport for free, but pay nothing.

Caladan Sell with 5 bsol per Mton.
Kaitain Sell with 6 bsol per Mton.
Giedi Prime Sell with 5 bsol per Mton.

With these prices, the Ixians get the revenue of 153 bsol, which just happens to be the same as the optimal shipping costs for the Spacing Guild. Of course, this is no coincidence. Also, it is quite curious that Tleilax should give its spice away for transportation for free!

9.12 Exercise (Spice Must Flow with No-Ships)
Solution 9.11 works only with the very special case in question. Make a GNU Octave function dualtranssolver that works like the function transsolver, but solves dual transportation problems. Solve Problem 9.9 by using dualtranssolver.

Hint: The easiest way to code the dualtranssolver is to extend transsolver to solve both the primal and the dual problem.

9.13 Exercise (More Spice)
More spice was found in Arrakis: there is now 23 Mtons there. Because of this excess supply, the planet Chapterhouse was promised 8 Mtons of spice. Transportation costs from Arrakis and Tleilax were negotiated with the Spacing Guild. The result is stated in the following transportation table:

Heighliner Spice Transportation					
Source/Destination	Caladan	Kaitain	Giedi Prime	Chapterhouse	**Supply**
Arrakis	3 bsol	4 bsol	6 bsol	**3** bsol	**23** Mton
Tleilax	5 bsol	7 bsol	5 bsol	**2** bsol	20 Mton
Demand	17 Mton	8 Mton	10 Mton	**8** Mton	

Solve the optimal transportation schedule for the Spacing Guild. Also, solve the optimal Ixian counteroffer. (See Problem 9.1 and Problem 9.9 for reference).

Chapter 10

More Transportation Problems

In this chapter we continue to study transportation problems. In the first section we study a generalization of a transportation problem called the transshipment problem. In the second section we consider a (almost) special case of a transportation problem called the assignment problem.

Again it should be noted that we use LP as a hammer, and everything is a nail to us. There are other, more efficient, ways of solving the transshipment problems and the assignment problems than the LP formulation. Indeed, for the assignment problems there is a very efficient and elegant algorithm called the **Hungarian algortihm**. Readers interested in the Hungarian algorithm can just google for it, or check the old ORMS1020 notes (page 147) here: www.uwasa.fi/∼tsottine/or_with_octave/or_with_octave.pdf

We will not consider dual transshipment or dual assignment problems. It is left for the readers' curiosity to find out what they are.

Transshipment Problem

In a **transshipment problem** one has to ship goods from sources to destinations as in a transportation problem, but in addition to the sources and the destinations there are **transshipment points** at one's disposal. So, one can ship goods directly from sources to destinations, or one can use transshipment points in the shipping.

We will consider only the case of one transshipment point. The case of multiple transshipment points is similar, but messy, especially if transportation between transshipment points is allowed.

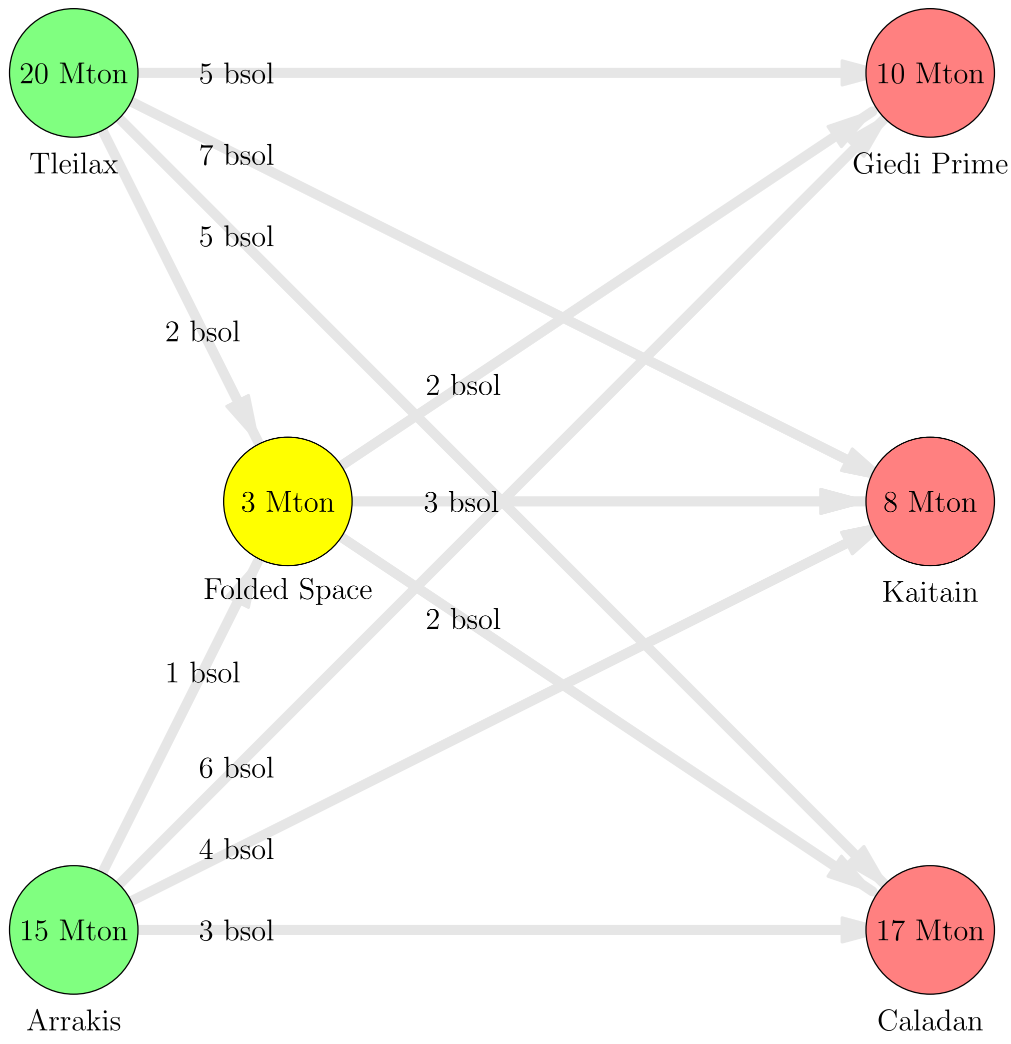
At first sight the transshipment problems are network problems. At least, they seem to be much more complicated than transportation problems. It turns out, however, that we can express transshipment problems as transportation problems. So the generalization from transportation problems to transshipment problems is in theory no generalization at all. In practice, it can be tedious at times.

Our key problem in this section is the transportation problem 9.1 with an added transshipment point.

10.1 Problem (Spice with Folded Space)
The Spacing Guild transports Spice Melange from Arrakis and Tleilax with its heighliners. The spice is transported to Caladan, Kaitain, and Giedi Prime; as explained in Problem 9.1.

The Spacing Guild has built a secret warehouse in the folded space. The Folded Space warehouse has capacity of 3 Mton. The Spacing Guild can now transport the spice directly or it can use its Folded Space warehouse as a transshipment point. The demands, supplies, and route costs are given in the graph below.

What is the optimal transportation schedule for the Spacing Guild?



The general idea of modeling transshipment problems as transportation problems is to understand that each **transshipment point is both a source and a destination**. So, each transshipment point has **two duplicates**: one on the source side and one on the destination side.

The costs for shipping to (destination duplicate) and from (source duplicate) a transshipment point are given in the problem.

The cost of shipping from the source duplicate to the destination duplicate is obviously zero.

The only remaining question is to figure out what are are the supplies for transshipment points (for their supply side duplicates), and what are the demands of for the transshipment points (for their demand side duplicates). It is maybe a bit unintuitive, but **both the demand and the supply for each transshipment point is its capacity**. If the capacity of a transshipment point is unlimited, then both the supply and demand for the transshipment point can be taken as its total potential supply.

Let us explain a little bit why both the supply and demand for a transshipment point is its capacity. Suppose that the transshipment point is not used at all. Then it will satisfy its supply and demand itself, and this will cost nothing. Suppose then that some amount of goods are sent to the (destination duplicate) transshipment point. If this outside supply does not exceed the capacity of the transshipment point, then the remaining demand is provided from the supply side duplicate of the transshipment point. If the outside supply exceeds the transshipment point's capacity, then the remaining demand from the supply side duplicate from the transshipment point would be negative. This would violate the sign constraints of the transportation problem. I hope you get the picture. If not, please draw it. I mean it! Please redraw the picture above by duplicating the yellow circle to the source side (left) and the destination side (right), and fill in the supplies, demands, and the route costs.

Here is the transportation formulation of the transshipment Problem 10.1:

Transshipment with Folded Space

Source/Destination	Caladan	Kaitain	Giedi Prime	Folded Space	**Supply**
Arrakis	3 bsol	4 bsol	6 bsol	1 bsol	15 Mton
Tleilax	5 bsol	7 bsol	5 bsol	2 bsol	20 Mton
Folded Space	2 bsol	3 bsol	2 bsol	0 bsol	3 Mton
Demand	17 Mton	8 Mton	10 Mton	3 Mton	

10.2 Solution (Spice with Folded Space)

We solve Problem 10.1 by modeling it as a transportation problem as figured out in the table above, and then use the home-made function transsolver. Below is the script m-file foldedspace.m that solves the problem. It can be downloaded from http://lipas.uwasa.fi/~tsottine/spicy_or/foldedspace.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Folded Space, solution with transsolver.
%% (Problem 10.1 from Linear Programming with Spice)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Original data without Folded Space warehouse
s = [15 20]';                    %% Supplies in Mtons.
d = [17  8 10]';                 %% Demands in Mtons.
C = [                            %% Route costs bsol per Mton.
3  4  6;
5  7  5
];
```

```
19
20 %% Inclusion of Folded Space warehouse
21 s = [s; 3];                    %% Folded Space supply duplicate capacity.
22 d = [d; 3];                    %% Folded Space demand duplicate capacity.
23 C = [C [1;2]];                 %% Route costs into Folded Space.
24 C = [C; [2 3 2 0]];            %% Route costs out of Folded Space.
25
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
27 %% Solve with transsolver and print out the solution.
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29
30 [X, cost] = transsolver(s,d,C)
31
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33 %% Test result:
34 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35 %%>> foldedspace
36 %%X =
37 %%
38 %%    10     5     0     0
39 %%     7     0    10     3
40 %%     0     3     0     0
41 %%
42 %%cost = 150
```

The workings of foldedspace.m should be quite clear at this point of readers' education. The only new thing here is that we decided ot emphasize the role of the Folded Space warehouse by first declaring the data without the warehouse in the lines 13–18 and then add the data related to the warehouse in the lines 21–24. Indeed, in the lines 21–22 both the supply and demand vectors are added on component: the Folded Space capacity. In the line 23, we add one column to the cost matrix giving the costs to the warehouse. In the line 24, twe add one row to he cost matrix giving the costs out of the Folded Space warehouse.

The **solution** is listed in the Test result section. The optimal transshipment schelude is X and the optimal transshipment cost is cost. For the readers' convenience we show below the optimal transshipment schedule with the problem data. (We have suppressed the units for marginal reasons.)

Transshipment with Folded Space
(Optimal Transshipment Schedule)

Source/Destination	Caladan	Kaitain	Giedi Prime	Folded Space	**Supply**
Arrakis	3 (10)	4 (5)	6 (0)	1 (0)	15
Tleilax	5 (7)	7 (0)	5 (10)	2 (3)	20
Folded Space	2 (0)	3 (3)	2 (0)	0 (0)	3
Demand	17	8	10	3	

In particular, we can see that the Folded Space warehouse is used in full capacity by the source Tleilax and all the spice in the warehouse is transported to Kaitain. The total price is 150 bsol. So, the warehouse decreased the total shipping cost by $153 - 150 = 3$ bsol (compare with Problem 9.1 and its Solution 9.7).

10.3 Exercise (Folded Space with Unlimited Capacity)
Consider Problem 10.1. The Spacing Guild is thinking of expanding the Folded Space warehouse to have unlimited capacity. How much should this improvement cost at most?

10.4 Exercise (transshipsolver)
Write a GNU Octave function transshipsolver that returns an optimal transshipment schedule and the optimal transshipment cost for a general transshipment problem with **one** transshipment point. The input parameters of the fuction transshipsolver should be s,d,C as in the associated transportation problem, and cap for the capacity of the transshipment point, Cin for the transportation costs into the transshipment point and Cout for the transportation costs out of the transshipment point. Test with Problem 10.1 that your function works.

Hint: A good starting point is to modify the script m-file foldedspace.m and to use the function transsolver.

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. In an assignment problem one has to assign each worker to each task (or vice versa) in a most efficient way.

Problem 10.5 below is a typical assignment problem.

10.5 Problem (Ghola Production)
Tleilaxu Master Scytale needs to build 4 different types of Gholas in 4 different types of Axlotl tanks. Each tank can build each type of gholas, except tank 1, that can only produce gholas of types 1–3. The Axlotl tanks have different setup times for different gholas given in the table below:

	Task			
Worker	Ghola 1	Ghola 2	Ghola 3	Ghola 4
Tank 1	14	5	8	–
Tank 2	2	12	6	5
Tank 3	7	8	3	9
Tank 4	2	4	6	10

Master Scytale wants to minimize the total setup time. What should he do?

If you think about it a while, you see that 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.

Let us build the LP representing Problem 10.5 by using Algorithm 5.2.

The key point in modeling the ghola production problem 10.5 is to find out the **decision variables** — everything else is easy after that. So what are the decisions Tleilaxu Master Scytale must make? He must choose which Axlotl tank is assigned to build which ghola. 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

$$X_{ij} \quad = \quad \begin{cases} 1 & \text{if tank } i \text{ is assigned to meet the demands of ghola } j, \\ 0 & \text{if tank } i \text{ is \textbf{not} assigned to meet the demands of ghola } j. \end{cases}$$

In the above we explained the decision variables X_{ij} in a funny way in order to make the link to a transportation problem. A more natural way of explaining the decision variables is of course: X_{ij} is an **indicator** of the claim "Axlotl tank i is assigned to build ghola j". The indicator of any claim is 1 if the claim is true, and 0 otherwise.

The rest of the steps in Algorithm 5.2 are now relatively straightforward.

The **objective** is to minimize the total setup time. With our binary variables we can write the total setup time as

$$\begin{aligned} z \;=\; & 14X_{11} + \;\, 5X_{12} + 8X_{13} + MX_{14} \\ & +2X_{21} + 12X_{22} + 6X_{23} + \;\, 5X_{24} \\ & +7X_{31} + \;\, 8X_{32} + 3X_{33} + \;\, 9X_{34} \\ & +2X_{41} + \;\, 4X_{42} + 6X_{43} + 10X_{44} \end{aligned}$$

Note that there will be a lot of zeros in the objective function above. Indeed, only 4 of the 16 decision variables will be 1's, the rest will be 0's. Also note the **big-M trick**. Since the setup where $X_{14} = 1$ is not allowed, we make it so expensive (choose big M), that in the solution we will have $X_{14} = 0$.

What about the **constraints** for ghola production? First, we have to ensure that each Axlotl tank is assigned to a ghola. This will give us the **supply constraints**

$$\begin{array}{ccccccccc} X_{11} & + & X_{12} & + & X_{13} & + & X_{14} & = & 1 \\ X_{21} & + & X_{22} & + & X_{23} & + & X_{24} & = & 1 \\ X_{31} & + & X_{32} & + & X_{33} & + & X_{34} & = & 1 \\ X_{41} & + & X_{42} & + & X_{43} & + & X_{44} & = & 1 \end{array}$$

Second, we have to ensure that each ghola is produced, i.e., each ghola has an Axlotl tank assigned to it. This will give us the **demand constraints**

$$\begin{array}{ccccccccc} X_{11} & + & X_{21} & + & X_{31} & + & X_{41} & = & 1 \\ X_{12} & + & X_{22} & + & X_{32} & + & X_{42} & = & 1 \\ X_{13} & + & X_{23} & + & X_{33} & + & X_{43} & = & 1 \\ X_{14} & + & X_{24} & + & X_{34} & + & X_{44} & = & 1 \end{array}$$

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 Problem 10.5:

$$\begin{array}{lll} & \min z = & \begin{aligned} & 14X_{11} + \;\, 5X_{12} + 8X_{13} + MX_{14} \\ & +2X_{21} + 12X_{22} + 6X_{23} + \;\, 5X_{24} \\ & +7X_{31} + \;\, 8X_{32} + 3X_{33} + \;\, 9X_{34} \\ & +2X_{41} + \;\, 4X_{42} + 6X_{43} + 10X_{44} \end{aligned} \\ \\ & \text{s.t.} & \begin{array}{ccccccccc} X_{11} & + & X_{12} & + & X_{13} & + & X_{14} & = & 1 \quad \text{(Tanks)} \\ X_{21} & + & X_{22} & + & X_{23} & + & X_{24} & = & 1 \\ X_{31} & + & X_{32} & + & X_{33} & + & X_{34} & = & 1 \\ X_{41} & + & X_{42} & + & X_{43} & + & X_{44} & = & 1 \end{array} \\ (10.6) & & \\ & & \begin{array}{ccccccccc} X_{11} & + & X_{21} & + & X_{31} & + & X_{41} & = & 1 \quad \text{(Gholas)} \\ X_{12} & + & X_{22} & + & X_{32} & + & X_{42} & = & 1 \\ X_{13} & + & X_{23} & + & X_{33} & + & X_{43} & = & 1 \\ X_{14} & + & X_{24} & + & X_{34} & + & X_{44} & = & 1 \end{array} \\ \\ & & X_{ij} = 0 \qquad \text{or} \qquad X_{ij} = 1 \end{array}$$

In (10.6) we have **binary constraints** $X_{ij} = 0$ or $X_{ij} = 1$ for the decision variables. So, at first sight it seems that the program (10.6) is not a linear one. Indeed, it is a special case of an integer program (IP) called **binary integer program** (BIP). 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 (usually) get an optimal solution where the decisions X^*_{ij} are either 0 or 1. Hence, the program (10.6) is a linear one, i.e., it is an LP. Or, to be more precise, the program (10.6) and its linear relaxation, or **LP relaxation**

$$
\begin{array}{lllllllllll}
\text{min } z = & 14X_{11} + 5X_{12} + 8X_{13} + MX_{14} \\
& +2X_{21} + 12X_{22} + 6X_{23} + 5X_{24} \\
& +7X_{31} + 8X_{32} + 3X_{33} + 9X_{34} \\
& +2X_{41} + 4X_{42} + 6X_{43} + 10X_{44} \\
\\
\text{s.t.} & X_{11} + X_{12} + X_{13} + X_{14} & = & 1 & \text{(Tanks)} \\
& X_{21} + X_{22} + X_{23} + X_{24} & = & 1 \\
& X_{31} + X_{32} + X_{33} + X_{34} & = & 1 \\
& X_{41} + X_{42} + X_{43} + X_{44} & = & 1 \\
\\
& X_{11} + X_{21} + X_{31} + X_{41} & = & 1 & \text{(Gholas)} \\
& X_{12} + X_{22} + X_{32} + X_{42} & = & 1 \\
& X_{13} + X_{23} + X_{33} + X_{43} & = & 1 \\
& X_{14} + X_{24} + X_{34} + X_{44} & = & 1 \\
\\
& \qquad\qquad\qquad\quad X_{ij} & \geq & 0
\end{array}
\tag{10.7}
$$

are equivalent. Equivalence of programs means that they have the same optimal decision and the same optimal objective value. Finally, we note that the programs (10.6) and (10.7) are equivalent to the following transportation problem:

$$
\begin{array}{lllllllllll}
\text{min } z = & 14X_{11} + 5X_{12} + 8X_{13} + MX_{14} \\
& +2X_{21} + 12X_{22} + 6X_{23} + 5X_{24} \\
& +7X_{31} + 8X_{32} + 3X_{33} + 9X_{34} \\
& +2X_{41} + 4X_{42} + 6X_{43} + 10X_{44} \\
\\
\text{s.t.} & X_{11} + X_{12} + X_{13} + X_{14} & \leq & 1 & \text{(Tanks are supplies)} \\
& X_{21} + X_{22} + X_{23} + X_{24} & \leq & 1 \\
& X_{31} + X_{32} + X_{33} + X_{34} & \leq & 1 \\
& X_{41} + X_{42} + X_{43} + X_{44} & \leq & 1 \\
\\
& X_{11} + X_{21} + X_{31} + X_{41} & \geq & 1 & \text{(Gholas are demands)} \\
& X_{12} + X_{22} + X_{32} + X_{42} & \geq & 1 \\
& X_{13} + X_{23} + X_{33} + X_{43} & \geq & 1 \\
& X_{14} + X_{24} + X_{34} + X_{44} & \geq & 1 \\
\\
& \qquad\qquad\qquad\quad X_{ij} & \geq & 0
\end{array}
\tag{10.8}
$$

10.9 Exercise (Ghola Production)
Solve Problem 10.5.

Hint: Use the LP interpretation (10.8) and the function `transsolver`.

10.10 Exercise (asssolver)
Write a GNU Octave function asssolver that returns an optimal setup and setup cost for an assignment problem. Test with Problem 10.5 that your function works.

Hint: A good starting point is to use the function transsolver.

Chapter 11

Data Envelopment Analysis

Data Envelopment Analysis (DEA) is a method of determining efficiencies of units (the so-called Decision Making Units, or DMU's). DEA is a data-driven approach in which the efficiencies are calculated solely on the basis of the inputs and outputs of the different DMU's.

We give only a very brief explanation of DEA. For more information the reader is referred to the notes www.uwasa.fi/∼tsottine/or_with_octave/or_with_octave.pdf Chapter 6.

Primal Approach

11.1 **Problem (Imperial Universities)**
God-emperor Leto II has decided to give special imperial status to universities that are efficient. Four universities, University of Caladan, University of Giedi Prime, University of Ix, and University of Kaitain have applied for the special imperial status. To assess the efficiency, god-emperor decided to consider the inputs and the outputs of the universities be:

Inputs: solaris, students, professors, and spice melange.
Outputs: suk doctors, mentats, and Ixian technology.

The inputs and the outputs of the applicant universities are given in the following table:

	Inputs				**Outputs**		
DMU	Solaris	Students	Profs	Spice	Suk Drs	Mentats	Ixtech
U Caladan	9 msol	12 500	1 300	3 kg	9 500	500	12 pats
U Giedi Prime	8 msol	10 600	700	2 kg	9 000	1 100	510 pats
U Ix	2 msol	0	1 100	72 kg	0	0	250 pats
U Kaitain	27 msol	47 200	2 520	5 kg	9 200	1 200	0 pats

The god-emperor decided to compare only the applicants to each others and give those that are 100 % efficient the special imperial status. Which universities should be given the status?

Before going into analysis, let us note that the data of the problem can be represented by two matrices: the **input matrix X** and the **output matrix Y**. We can choose to put the universities (henceforth called the **DMU's** or **Decision Making Units**) in the rows or in the columns of the matrices $\mathbf{X}$ and $\mathbf{Y}$. We choose the confusing way of putting them into columns (not as rows as in Problem 11.1). Thus

$$\begin{aligned} X_{io} &= \text{the amount of the input } i \text{ for the DMU}_o, \\ Y_{jo} &= \text{the amount of the output } j \text{ for the DMU}_o. \end{aligned}$$

Thus the data of Problem 11.1 is given by the matrices

$$\mathbf{X} = \begin{bmatrix} 9 & 8 & 2 & 27 \\ 12500 & 10600 & 0 & 47200 \\ 1300 & 700 & 1100 & 2520 \\ 3 & 2 & 72 & 5 \end{bmatrix}$$

and

$$\mathbf{Y} = \begin{bmatrix} 9500 & 9000 & 0 & 9200 \\ 500 & 1100 & 0 & 1200 \\ 12 & 510 & 250 & 0 \end{bmatrix}$$

Let us consider the efficiency of the University of Caladan, so the DMU under consideration is $\text{DMU}_o = \text{DMU}_1$.

The most obvious way to measure efficiencies (let us call them $\boldsymbol{\theta}$ and in particular θ_o for the DMU under consideration) with multiple inputs and outputs is to use weighted rations: Informally

$$\boldsymbol{\theta} = \frac{\text{weighted outputs}}{\text{weighted inputs}}.$$

In the case of U Caladan, which is our DMU_o now, this would mean that

$$\begin{aligned} \theta_o &= \frac{9\,500u_1 + 500u_2 + 12u_3}{9v_1 + 12\,500v_2 + 1\,300v_3 + 3v_4} \\ &= \frac{\sum_i u_i Y_{io}}{\sum_j v_j X_{jo}} \\ &= \frac{\mathbf{u}'\mathbf{Y}_{\bullet o}}{\mathbf{v}'X_{\bullet o}} \end{aligned}$$

Here the $\mathbf{u}$ and $\mathbf{v}$ are the weights for the outputs and for the inputs. Now the problem is: how to choose the weights?

The most generous way to choose the weight is to let the DMU under consideration to choose them. The DMU under consideration will then of course choose them in the way that makes it look good. Thus DMU_o is given the maximization problem

$$\max \theta_o = \frac{\mathbf{u}'\mathbf{Y}_{\bullet o}}{\mathbf{v}'\mathbf{X}_{\bullet o}}$$

This is a **fractional objective** function, not a linear one. This may be a problem, since GLPK can only solve linear problems. But since we are dealing with fractions, we can always scale the solution so that the denominator is 1. Thus the fractional optimization becomes a linear one with a constraint:

$$\begin{array}{lrcl} \max & \theta_o = \mathbf{u}'\mathbf{Y}_{\bullet o} & & \\ \text{s.t.} & \mathbf{v}'X_{\bullet o} & = & 1. \end{array}$$

But we need more constraints! Indeed the poblem above would obviously have unbounded solution. The clever trick here is that **all the other DMU's will be weighted with our weights and no-one should have more than 100% efficiency.** Thus we end up with the following near LP for the DMU$_o$ when there are n DMU's in total:

$$\begin{array}{lrcl} \max & \theta_o = \mathbf{u}'\mathbf{Y}_{\bullet o} & & \\ \text{s.t.} & \mathbf{v}'\mathbf{X}_{\bullet o} & = & 1 \\ & \frac{\mathbf{u}'\mathbf{Y}_{\bullet 1}}{\mathbf{v}'\mathbf{X}_{\bullet 1}} & \le & 1 \\ & \frac{\mathbf{u}'\mathbf{Y}_{\bullet 2}}{\mathbf{v}'\mathbf{X}_{\bullet 2}} & \le & 1 \\ & & \vdots & \\ & \frac{\mathbf{u}'\mathbf{Y}_{\bullet n}}{\mathbf{v}'\mathbf{X}_{\bullet n}} & \le & 1 \\ & \mathbf{u},\mathbf{v} & \ge & 0 \end{array} .$$

The constrants for the other DMUs seem fractional, not linear. But this is not a problem. Indeed, we can write the near LP above as

$$\begin{array}{lrcl} \max & \theta_o = \mathbf{u}'\mathbf{Y}_{\bullet o} & & \\ \text{s.t.} & \mathbf{v}'\mathbf{X}_{\bullet o} & = & 1 \\ & \mathbf{u}'\mathbf{Y}_{\bullet 1} - \mathbf{v}'\mathbf{X}_{\bullet 1} & \le & 0 \\ & \mathbf{u}'\mathbf{Y}_{\bullet 2} - \mathbf{v}'\mathbf{X}_{\bullet 2} & \le & 0 \\ & & \vdots & \\ & \mathbf{u}'\mathbf{Y}_{\bullet n} - \mathbf{v}'\mathbf{X}_{\bullet n} & \le & 0 \\ & \mathbf{u},\mathbf{v} & \ge & 0 \end{array} . \tag{11.2}$$

This is an LP with variable $\mathbf{x} = [\mathbf{u}'\ \mathbf{v}']'$. The objective comes from $\mathbf{Y}$ and the technology matrix is a combitation of $\mathbf{X}$ and $\mathbf{Y}$. To find the precise LP formulation is the next exercise.

11.3 Exercise (DEA LP)
Find $\mathbf{c}$, $\mathbf{A}$ and $\mathbf{b}$ such that the problem (11.2) is a standard from LP with the decision variable $\mathbf{x} = [\mathbf{u}'\ \mathbf{v}']'$.

Hint: Look at the code impuni.m, especially the lines 55–58.

11.4 Solution (Imperial Universities)
Here is the script m-file impuni.m that solves Problem 11.1. The file impuni.m is written in a way that is easily extended to any input and output matrices $\mathbf{X}$ and $\mathbf{Y}$. The script file can be downloaded from www.uwasa.fi/∼tsottine/spicy_or/impuni.m

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%
3 %% Imperial universities DEA problem 11.1
4 %%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data: X is inputs and Y is outputs  (note that DMU's are in the columns:
%% hence the transposes).  Also, some shorthands.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X = [
          9      12500    1300      3;     %% U Caladan
          8      10600     700      2;     %% U Giedi Prime
          2          0    1100     72;     %% U Ix
         27      47200    2520      5      %% U Kaitain
];
%% solaris  Students  Profs Spice

nDMU = columns(X);       %% Number of DMUs.
oDMU = zeros(1,nDMU);    %% Zeros for DMUs.

nin = rows(X);           %% Number of inputs.
oin = zeros(1,nin);      %% Zeros for inputs.

Y = [
       9500       500       12;     %% U Caladan
       9000      1100      510;     %% U Giedi Prime
          0         0      250;     %% U Ix
       9200      1200        0      %% U Kaitain
];
%% Suk Drs  Mentats  Ixtech

nout = rows(Y);          %% Number of outputs.
oout = zeros(1,nout);    %% Zeros for outputs.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculate the efficiencies in a for loop for each DMU.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% ctype and vtype are the same for each DMU.
ctype = "S";
for i = 1:nDMU
    ctype = [ctype "U"];
end

vtype = "";
for j = 1:(nout+nin)
  vtype = [vtype "C"];
end

%% The for loop itself
for o = 1:nDMU
    c = [Y(:,o)'  oin]';                                      %% Objective.
    A = [ oout X(:,o)'; Y' -X'];                              %% Technology.
    b = [1 oDMU]';                                            %% Bounds.
    [tmp, theta(o)] = glpk(c,A,b,[],[],ctype,vtype,-1);      %% The BEEF!
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Efficiencies printing.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
theta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Test output:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%>> impuni
%%theta =
%%
%%    0.9383    1.0000    1.0000    0.4364
```

We see that the efficiencies are

University of Caladan 94%
University of Giedi Prime 100%
University of Ix 100%
University of Kaitain 44%

So, University of Giedi Prime and University of Ix will be given special imperial status.

11.5 Exercise
Make a function dea(X,Y) that takes in the input matrix $\mathbf{X}$ and the ouput matrix $\mathbf{Y}$ and returns the efficiencies of the associated DMUs.

Hint: Start with the code impuni.m.

Dual Approach

We know that every LP has a dual LP. So does the DEA LP. In the primal approach for DEA each DMU found weights for their inputs and outputs so that their efficiencies were maximized. The constraints were related to the other DMU's. In the dual approach the decision variables are related to the (other) DMU's. It turns out that these decision variables, let us call them $\boldsymbol{\lambda} = [\lambda_1 \ \cdots \ \lambda_n]'$ for the n DMU's for each DMU$_o$, can be considered to form a **virtual DMU**

$$\mathrm{DMU}_{\mathrm{virtual}} = \sum_{k=1}^{n} \lambda_k \mathrm{DMU}_k.$$

This virtual DMU will have the same **business mix** than DMU$_o$ and it will be 100% efficient.

The virtual coefficients $\boldsymbol{\lambda}$ and the efficiency ϑ for DMU$_o$ come from the following dual DEA problem:

$$\begin{array}{llcl} \min & \vartheta & & \\ \text{s.t.} & \mathbf{Y}\boldsymbol{\lambda} & \geq & \mathbf{Y}_{\bullet o} \\ & \mathbf{X}\boldsymbol{\lambda} & \leq & \vartheta \mathbf{X}_{\bullet o} \\ & \boldsymbol{\lambda}, \vartheta & \geq & 0 \end{array} \tag{11.6}$$

11.7 Exercise (Dual Imperial Universities)
Solve the Imperial Univiersities problem 11.1 by using the dual LP formulation (11.6).

11.8 Exercise (Dual DEA LP)
Derive the dual DEA LP (11.6) from the primal DEA LP (11.2).

Chapter 12

A Look Under the Hood

In this chapter we will briefly look under the hood of glpk. That is, we explain a little bit how and why does it work; and as true mentats we adhere to the ban of thinking machines and do everything by hand in this chapter.

We give only a very brief explanation. For more information the reader is referred to the notes www.uwasa.fi/~tsottine/or_with_octave/or_with_octave.pdf Chapters 3 and 4.

The Truth is in the Corners

Recall that every LP can be given in standard form

$$\begin{array}{llcl} \max & z = \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} . \tag{12.1}$$

The constraints

$$\begin{array}{rcl} \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array}$$

define the so-called **feasible region**, i.e. the region of those points $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]'$ in an n-dimensional space where the possible solutions $\mathbf{x}$ of the maximization problems lie. It can be shown that the feasible region is an n-dimensional polygon. Moreover, it can be shown that **the optimal values $\mathbf{x}^*$ of the LP can be found in the corners of the feasible region**.

To identify the corners of the feasible region, it is convenient to rewrite the standard from LP (12.1) in the following so-called **slack form**

$$\begin{array}{llcl} \max & z = \mathbf{c}'\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} + \mathbf{s} & = & \mathbf{b} \\ & \mathbf{x}, \mathbf{s} & \geq & \mathbf{0} \end{array} . \tag{12.2}$$

Here the **slack** $\mathbf{s} = [s_1\ s_2\ \cdots\ s_m]'$ tells how much there is slack in each constraint. Note that taking $\bar{\mathbf{x}} = [\mathbf{x}'\ \ \mathbf{s}']'$ to be the augmented decision variables, we can write the slack form as

$$\begin{array}{llcl} \max & z = [\mathbf{c}'\ \mathbf{0}]\bar{\mathbf{x}} & & \\ \text{s.t.} & [\mathbf{A}\ \mathbf{I}]\bar{\mathbf{x}} & = & \mathbf{b} \\ & \bar{\mathbf{x}} & \geq & \mathbf{0} \end{array} .$$

To find the corners of the feasible region on simply omits the non-negativity constraints and solves the system of equations

$$[\mathbf{A}\ \mathbf{I}]\bar{\mathbf{x}} = \mathbf{b}. \tag{12.3}$$

Since there are $n+m$ variables and only m equations, the system (12.3) has many solutions. Indeed, it has $\binom{m+n}{m}$ solutions. So, for example an LP with 3 decision variables and 4 constraints yields a system with

$$\binom{7}{4} = \frac{7!}{4!3!} = 35$$

solutions. Thus, the feasible region of the original LP has 35 potential corners.

All the corners of the original LP's feasible region are some solutions of the system (12.3). They are found by choosing m components of $\bar{\mathbf{x}}$ to be so-called **basic variables** (**BV**) and setting the remaining components to be zero. There remaining zero components are called **non-basic variables** (**NBV**). For each m BV's the solutions $\bar{\mathbf{x}}$ are called **basic feasible solutions** (**BFS**) if they are also feasible solutions. In other words, the corners of the feasible region are basic feasible solutions. Yet in other words a solution $\bar{\mathbf{x}}$ of the system (12.3) that also satisfies the non-negativity constraints $\bar{\mathbf{x}} \geq \mathbf{0}$ is a BFS of the original LP and consequently a corner of the feasible region.

12.4 Example (Checking Corners)
Let us Consider the LP

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

This LP is in standard form, so its slack form is easy to find:

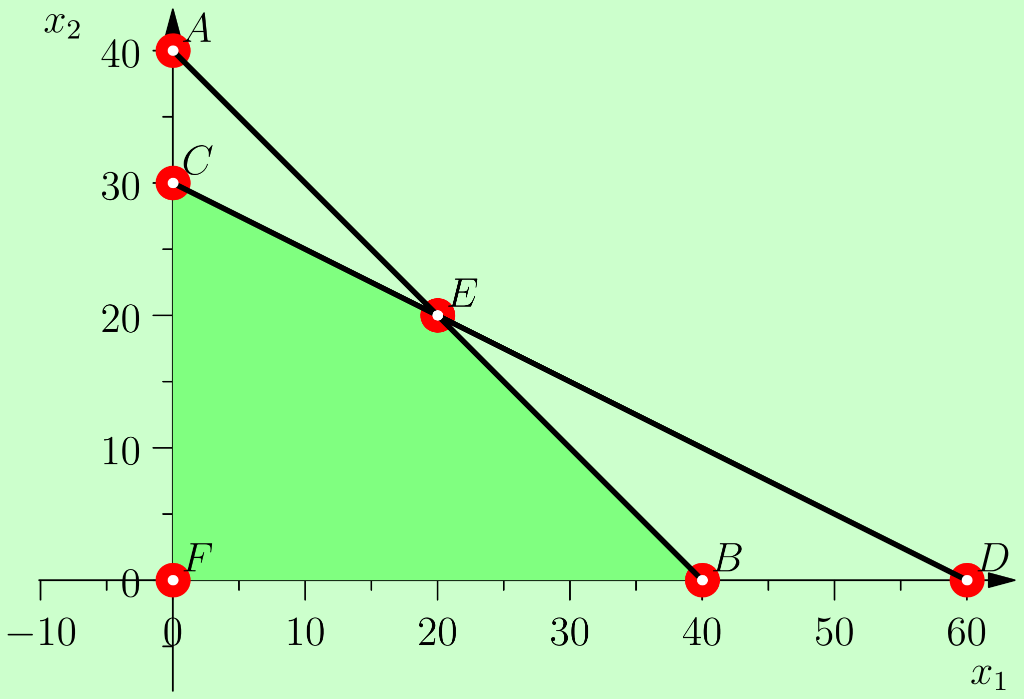
$$\begin{array}{lrcrcrcrcl} \max z & = 3x_1 & + & 4x_2 & & & & & & \\ \text{s.t.} & x_1 & + & x_2 & + & s_1 & & & = & 40 \\ & x_1 & + & 2x_2 & & & + & s_2 & = & 60 \\ & & & & & & x_1, x_2, & s_1, s_2 & \geq & 0 \end{array}$$

Let us solve this slack form by checking all the potential corners of the feasible region.

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

From the next table (or the next picture) we read that the optimal decision is $x_1 = 20$, $x_2 = 20$ with the slacks $s_1 = 0$, $s_2 = 0$. The corresponding optimal value is $z = 140$.

BV	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	No	–	A
x_2, s_1	$0 + x_2 + s_1 = 40$ $0 + 2x_2 + 0 = 60$	0	30	10	0	Yes	120	C
x_1, s_2	$x_1 + 0 + 0 = 40$ $x_1 + 0 + s_2 = 60$	40	0	0	20	Yes	120	B
x_1, s_1	$x_1 + 0 + s_1 = 40$ $x_1 + 0 + 0 = 60$	60	0	-20	0	No	–	D
x_1, x_2	$x_1 + x_2 + 0 = 40$ $x_1 + 2x_2 + 0 = 60$	20	20	0	0	Yes	140	E



12.5 Exercise (Check the Corners)
Solve the LP

$$
\begin{array}{rrcrcrcr}
 & & & \min z & = & -2x_1 + 3x_2 & & \\
\text{s.t.} & 1 & \leq & x_1 & + & x_2 & \leq & 9 \\
 & & & 2x_1 & - & x_2 & \leq & 4 \\
 & 2 & \leq & 7x_1 & + & x_2 & \leq & 100 \\
 & & & & & x_2 & \geq & 0
\end{array}
$$

by checking the corners of its feasible region.

Hint: Transform the LP first into a standard form, and then into a slack form, and follow Example 12.4 above. Or, just draw a picture.

12.6 Remark (Combinatorial Curse)
It seems we now have a very simple algorithm for finding an optimum: Just check all the corners! And, indeed, this naive approach works well with such petty examples we have in this course. The problem with this brute force approach in practice is a manifestation of the **combinatorial curse**: an LP with n decision variables and m constraints has

$$\binom{n+m}{m} \quad = \quad \frac{(n+m)!}{n!\,m!}$$

corners (in the slack form system). So, an LP with 15 decision variables and 15 constraints has

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

corners. Suppose you have a computer that checks 1 000 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 two days for the computer to check all the 155 117 520 corners. You may think this is not a problem: maybe two days is not such a long time, and a problem with 15 decision variables is way bigger than anything you would encounter in the real life anyway. Well, think again! Two days is a long time if you need to update your optimal solution in a changing environment of, say, a stock exchange, and LP's with at 15 decision variables are actually rather small. Indeed, let us be a bit more realistic now: Consider a stock broker who has 50 stocks in her stock (pun intended). Suppose the broker has 50 constraints in selecting the stocks for her portfolio (not unreasonable) and a super-computer 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 portfolio would take 6.89×10^{44} years. The author doubts that even the universe can wait that long!

The bottom line: **Checking all the corners would take too long**.

Simplex Algorithm

By Remark 12.6 we cannot in practice check **all** the corners. So what do we do then. The natural idea is to start with one corner and then, if needed, go to a next, possibly better, corner, and continue until we hopefully hit the best corner. This is what the so-called **simplex algorithm** (and glpk) does in practice.

We will solve the following LP by hand by using the simplex algorithm:

$$
\begin{array}{lcrcrcrcr}
\max z & = & 60x_1 & + & 30x_2 & + & 20x_3 & & \\
\text{s.t.} & & 8x_1 & + & 6x_2 & + & x_3 & \le & 48 \\
 & & 4x_1 & + & 2x_2 & + & 1.5x_3 & \le & 20 \\
 & & 2x_1 & + & 1.5x_2 & + & 0.5x_3 & \le & 8 \\
 & & & & x_2 & & & \le & 5 \\
 & & & & & & x_1, x_2, x_3 & \ge & 0
\end{array}
\tag{12.7}
$$

This is a standard form LP, so it is easy to write its slack form:

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

Treating z as a "decision" variable we can go even further into an augmented slack form

$$\begin{array}{llcrcrcrcccccccl}
\max z & & & & & & & & & & & & & & & \\
\text{s.t.} & z & - & 60x_1 & - & 30x_2 & - & 20x_3 & & & & & & & = & 0 \\
 & & & 8x_1 & + & 6x_2 & + & x_3 & + & s_1 & & & & & = & 48 \\
 & & & 4x_1 & + & 2x_2 & + & 1.5x_3 & & & + & s_2 & & & = & 20 \\
 & & & 2x_1 & + & 1.5x_2 & + & 0.5x_3 & & & & & + s_3 & & = & 8 \\
 & & & & & x_2 & & & & & & & & + s_4 & = & 5 \\
 & & & & & & & & & & & \multicolumn{3}{r}{x_1,x_2,x_3,s_1,s_2,s_3,s_4} & \geq & 0
\end{array} \tag{12.8}$$

Now the idea is to look a corner (also called **bastic feasible solution**, BFS) that is defined by solving the system above by what is so-called **basic variables** (BV). The rest of the variables are called **non-basic variables** (NBV), and they are set to 0. So, there will be 4 (number of constraints) BV's and the rest 3 are set to zero as NBV's.

We see that the slack form is already solved for slacks. Thus, taking s_1, s_2, s_3, s_4 to be our first BV our **first simplex tableau** for (12.7), or equally for (12.8), is

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

In a very small nut-shell the simplex algorithm works now as follows:

If the simplex algorithm identifies the current simplex tableau to be solved with an optimal BFS there is nothing to be done any more, and the optimal solution can be read from the tableau with the given BV corresponding the BFS. If the current BFS is not optimal the simplex algorithm starts to look for a better BFS by changing the BV and then solving the tableau with respect to the new, hopefully better, BV. This is repeated until an optimal solution is found (if ever).

The criterion for optimality is:

The simplex tableau is optimal if there are no negative coefficients in the first row corresponding to the non-basic variables.

In the first simplex tableau above all the coefficients of NBV's x_1, x_2, x_3 are certainly negative. So, the first simplex tableau is not optimal.

Now we need to improve the simplex tableau. The improvement routing is a very critical part of the simplex algorithm. Indeed, so far the algorithm just starts with one BFS and checks its optimality. If the improvement routine that gives the next BFS is not good, we might end up looking for new BFSs for a very long time, or even forever! (This may still happen with the improvement routine we present here, but the probability of such a bad luck is tiny.)

The general idea of the improvement routine is to find the "best" variable in the NBV and the "worst" variable in the BV and then make a switch.

The entering variable is the one with the smallest coefficient in the first row.

The idea is that this way (we hope) to increase the RHS of the first row as much and as fast as possible. So, we find that the variable x_1 will enter the BV since it has the smallest coefficient -60.

The leaving BV will the one associated to the row that wins the ratio test (the smallest finite positive value is the winner)

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

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.

The ratio test gives

$$\begin{array}{lclcll}
\text{Row 2 limit for } x_1 & = & 48/8 & = & 6 & \\
\text{Row 3 limit for } x_1 & = & 20/4 & = & 5 & \\
\text{Row 4 limit for } x_1 & = & 8/2 & = & 4 & \\
\text{Row 5 limit for } x_1 & = & 5/0 & = & \infty & \text{No limit.}
\end{array}$$

So, Row 4 wins the ratio test. Hence s_3 is the leaving BV.

Now we have new BV: s_1, s_2, x_1, s_4 and an **unsolved simplex tableau**

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

Now we have to solve this Simplex tableau in terms of the BV. 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 by pivoting with respect to the BV that just entered. After some **tedious pivoting** we get the **solved simplex tableau**

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

This tableau is not yet optimal, since the NBV x_3 has negative coefficient -5. It is obvious that x_3 enters the BV. To find out the leaving variable we perform the ratio test:

$$\begin{array}{lcrcll}
\text{Row 2 limit for } x_3 & = & 16/(-1) & = & -16 & \text{No limit} \\
\text{Row 3 limit for } x_3 & = & 4/0.5 & = & 8 & \\
\text{Row 4 limit for } x_3 & = & 4/0.25 & = & 16 & \\
\text{Row 5 limit for } x_3 & = & 5/0 & = & \infty & \text{No limit}
\end{array}$$

So, row 3 wins the ratio test. Since s_2 is the BV of row 3, s_2 will leave. 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
1	1	0	15	-5	0	0	30	0	$z=$	240
2	0	0	0	-1	1	0	-4	0	$s_1=$	16
3	0	0	-1	0.5	0	1	-2	0	$x_3=$	4
4	0	1	0.75	0.25	0	0	0.5	0	$x_1=$	4
5	0	0	1	0	0	0	0	1	$s_4=$	5

which can be solved easily (but tediously) enough by pivoting. We obtain

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

and note that the tableau is indeed optimal! No need to continue table-dancing!

Finally let us read the solution and some extra results from the optimal simplex tableau. The optimal value is clearly visible: $z = 280$. The optimal decision is $x_1 = 2$, $x_2 = 0$ and $x_3 = 8$. The value $x_2 = 0$ is not explicitly written, but it is implied to be 0, since x_2 is a NBV. Finally, we can read the reduced costs and shadow variables from the first row. The reduced costs are the coefficients of the decision variables and the shadow prices are the coefficients of the slacks in the row 1.

12.9 Exercise (Manual Labor)
Solve the following LP manually by using simplex algorithm

$$\begin{array}{lcrcrcrcr}
\max z & = & 30x_1 & + & 40x_2 & + & 10x_3 & & \\
\text{s.t.} & & 8x_1 & + & 6x_2 & + & x_3 & \leq & 40 \\
 & & 4x_1 & + & 2x_2 & + & 2x_3 & \leq & 25 \\
 & & 2x_1 & & & + & 5x_3 & \leq & 10 \\
 & & & & & & x_1, x_2, x_3 & \geq & 0
\end{array}.$$

12.10 Exercise (Simplex Fail)
Simplex algortihm presented here will fail if **b** has negative components in the standard from LP

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

Why is this and can you figure out a solution?