Chapter 1
Linear programming

We start by giving some examples of linear programs and how they are used in practice.

A healthy and low-priced diet

Imagine you are going on a long vacation and you need to buy food. Your objective is to buy food at minimum price, such that the daily needs of certain vitamins and energy are satisfied. There are three kinds of food. Carrots, white cabbage and oatmeal, each having a certain amount of Vitamin A, Vitamin C and energy per 100g serving. [1]

- 100g carrots contain: 3.5 mg Vitamin A, 6 mg Vitamin C, 50 kcal Energy
- 100g white cabbage contains: 0.1 mg Vitamin A, 30 mg Vitamin C, 70 kcal Energy
- 100g Oatmeal contains: 0.02 mg Vitamin A, 0.04mg Vitamin C and 300 kcal Energy

The prices for 100g of the above are 1 CHF, 0.5 Chf and 3 CHF respectively. Your daily needs are

- Vitamin A: 0.75 mg
- Vitamin C: 0.5 mg
- Energy: 1500 kcal

Your goal is now to come up with the right mix of these dishes, such that all your needs in terms of energy, vitamin A, and vitamin C are satisfied and such that this mix is as cheap as possible.

This is done with a *linear program*, a central object of study in this course. We reserve variables x_1, x_2 and x_3 which is the amount of 100g units of carrots,

[1] Those are fantasy values. We are doing math and no dietary consulting ;)

cabbage and oatmeal respectively that we will eat each day. We want that the cost of a daily serving is minimized, in other words, we want to minimize the following linear function

$$\min 1 \cdot x_1 + 0.5 \cdot x_2 + 3 \cdot x_3.$$

Certain constraints have to be satisfied. The constraint, which tells us that we need at least $0.75mg$ of vitamin A is

$$3.5x_1 + 0.1x_2 + 0.02x_3 \geqslant 0.75.$$

The variables x_1, x_2 and x_3 have to be nonnegative, so all-together, we have to solve the following problem

$$\begin{array}{ll} \min & 1 \cdot x_1 + 0.5 \cdot x_2 + 3 \cdot x_3 \\ \text{subject to} & x_1 \geqslant 0 \\ & x_2 \geqslant 0 \\ & x_3 \geqslant 0 \\ & 3.5x_1 + 0.1x_2 + 0.02x_3 \geqslant 0.75 \\ & 6x_1 + 30x_2 + 0.04x_3 \geqslant 0.5 \\ & 50x_1 + 70x_2 + 300x_3 \geqslant 1500. \end{array}$$

Linear Programs

We use the following notation. For a matrix $A \in \mathbb{R}^{m \times n}$, $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ we denote the i-th row of A by a_i and the j-th column of A by a^j. With $A(i, j)$ we denote the element of A which is in the i-th row and j-th column of A. For a vector $v \in \mathbb{R}^m$ and $i \in \{1, \ldots, m\}$ we denote the i-th element of v by $v(i)$.

Definition 1.1. Let $A \in \mathbb{R}^{m \times n}$ be a matrix, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ be vectors and $I_{\geqslant}, I_{\leqslant}, I_{=} \subseteq \{1, \ldots, m\}$ and $J_{\geqslant}, J_{\leqslant} \subseteq \{1, \ldots, n\}$ be index sets. A *linear program (LP)* consists of

i) a linear *objective function*:

$$\begin{array}{c} \max c^T x \\ \text{or } \min c^T x \end{array}$$

ii) Linear *constraints*

$$\begin{array}{c} a_i^T x \geqslant b(i), i \in I_{\geqslant} \\ a_j^T x \leqslant b(j), j \in I_{\leqslant} \\ a_k^T x = b(k), k \in I_{=} \end{array}$$

iii) and *bounds on the variables*

$$\begin{array}{c} x(j) \geqslant 0,\ j \in J_{\geqslant} \\ x(j) \leqslant 0,\ j \in J_{\leqslant}. \end{array}$$

Notice that we can re-write the objective function $\min c^T x$ as $\max -c^T x$. Similarly, the constraints $a_i^T x \geqslant b(i), i \in I_{\geqslant}$ are equivalent to the constrains $-a_i^T x \leqslant -b(i), i \in I_{\geqslant}$. Also the constraints $a_k^T x = b(k), k \in I_{=}$ can be replaced by the constraints $a_k^T x \leqslant b(k)$, $-a_k^T x \leqslant -b(k)$, $k \in I_{=}$. A lower bound $x(j) \geqslant 0$ can be written as $-e_j^T x \leqslant 0$, where e_j is the j-th unit vector which has zeroes in every component, except for the j-th component, which is 1. Similarly an upper bound $x(j) \leqslant 0$ can be written as $e_j^T x \leqslant 0$.

All-together, a linear program as in Definition 1.1 can always be written as

$$\max\{c^T x\colon \widetilde{A}x \leqslant \widetilde{b}, x \in \mathbb{R}^n\}$$

with a suitable matrix $\widetilde{A} \in \mathbb{R}^{m \times n}$ and a suitable vector $\widetilde{b} \in \mathbb{R}^m$. This representation has a name.

Definition 1.2. A linear program is in *inequality standard form*, if it is of the form

$$\max\{c^T x\colon Ax \leqslant b, x \in \mathbb{R}^n\}$$

for some matrix $A \in \mathbb{R}^{m \times n}$ and some vector $b \in \mathbb{R}^m$.

Definition 1.3. A point $x^* \in \mathbb{R}^n$ is called *feasible*, if x^* satisfies all constraints and bounds on the variables. If there are feasible solutions of a linear program, then the linear program is called *feasible* itself. A linear program is *bounded* if there exists a constant $M \in \mathbb{R}$ such for all feasible $x^* \in \mathbb{R}^n$ $c^T x^* \leqslant M$, if the linear program is a maximization problem and $c^T x^* \geqslant M$, if the linear program is a minimization problem. A feasible solution x^* is an optimal solution if $c^T x^* \geqslant c^T y^*$ for all feasible y^* if the linear program is a maximization problem and $c^T x^* \leqslant c^T y^*$ if the linear program is a minimization problem.

We will see later that a feasible and bounded linear program has an optimal solution.

Two-variable linear programs

Two-variable linear programs can be solved graphically. Consider for example the linear program

$$\begin{array}{ll} \max x_1 + x_2 & \\ 2x_1 + 3x_2 & \leqslant 9 \\ 2x_1 + x_2 & \leqslant 5 \\ x_1, x_2 \geqslant 0. & \end{array}$$

Figure 1 depicts the feasible solutions as the gray area. The red vector is the objective vector $(1,1)$. This linear program is feasible and bounded. The optimal solution is the intersection of the two lines $2x_1 + x_2 = 5$ and $2x_1 + 3x_2 = 9$. This intersection is $x^* = (3/2, 2)$.

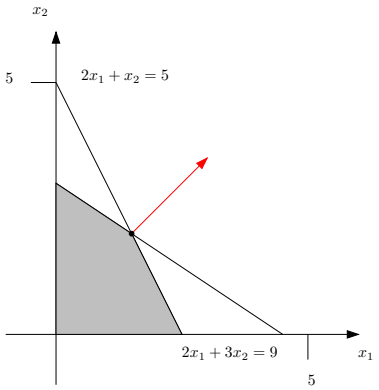

Fig. 1 A two-variable linear program

Fitting a line

The following is an example which is well known in statistics. Suppose that you measure points $(y_i, x_i) \in \mathbb{R}^2 i = 1, \ldots, n$ and you are interested in a linear function $y = a \cdot x + b$ that reflects the sample. One way to do that is by minimizing the expression

$$\sum_{i=1}^{n} (ax_i + b - y_i)^2, \tag{1}$$

where $a, b \in \mathbb{R}$ are the parameters of the line that we are looking for. The number $(ax_i + b - y_i)^2$ is the square of the vertical distance of the point x_i, y_i from the line $y = a\,x + b$.

Instead of using the method of least-squares, we could also minimize the following function, see also [3, Chapter 2.4],

$$\sum_{i=1}^{n} |ax_i + b - y_i|. \tag{2}$$

This objective has the advantage to be slightly more robust towards a outliers. How can we model this as a linear program. The trick is to use an extra variable h_i which models the absolute value of $ax_1 + b - y_i$.

$$\begin{array}{lll} \min \sum_{i=1}^{n} h_i & & \\ h_i & \geqslant & ax_i + b - y_i,\ i = 1,\ldots,n \\ h_i & \geqslant & -(ax_i + b - y_i),\ i = 1,\ldots,n \end{array} \tag{3}$$

The variables of this linear program are h_i, $i = 1,\ldots,n$, a and b. For a fixed $a \in \mathbb{R}$ and $b \in \mathbb{R}$ the optimal h_i's will be $h_i = |ax_i + b - y_i|$ since the objective minimizes the sum of the h_i's. If one of the was strictly larger than $|ax_i + b - y_i|$, then the objective could be improved by making it smaller.

Linear Programming solvers and modeling languages

We will demonstrate now how to use a modeling language for linear programming and a linear programming solver to find a fitting line, as described in Section 1 for the points

$$(1,3),(2.8,3.3),(4,2),(5.5,2.1),(6,.2),(7,1.3),(7.5,1),(8.5,0.8)$$

There are two popular formats for linear programming problems which are widely used by linear programming solvers, the *lp-format* and the *mps-format*. Both are not easy to read. To facilitate the modeling of a linear program, so-called modeling languages are used. We demonstrate the use of the popular open source modeling software called zimpl [2]. Below you see a way to model our fitting line linear program with zimpl:

```
set I := { 1 to 8};
param X[I] :=  <1> 1, <2> 2.8, <3> 4, <4> 5.5,
               <5> 6, <6> 7, <7> 7.5, <8> 8.5 ;
param Y[I] :=  <1> 3, <2> 3.3, <3> 2, <4> 2.1,
               <5> .2, <6> 1.3, <7> 1, <8> .8 ;
var h[I] >= -infinity <= infinity;
var a  >= -infinity <= infinity ;
var b  >= -infinity <= infinity ;

minimize cost: sum <i> in I: h[i];

subto c1: forall <i> in I:      h[i] >=   ( a * X[i] + b -Y[i]);
subto c2: forall <i> in I:      h[i] >= - ( a * X[i] + b -Y[i]);
```

Zimpl creates a linear program which is readable by linear programming solvers like QSopt or SoPlex. An optimal fitting-line w.r.t. the distance measure (2) is the line $y = -0.293333 \cdot x + 3.293333$. It is depicted in figure 1.

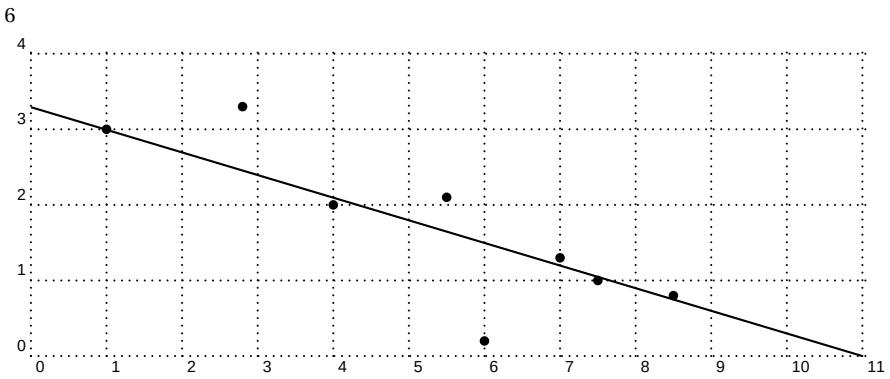

Fig. 2 A set of points $\{(1,3),(2.8,3.3),(4,2),(5.5,2.1),(6,.2),(7,1.3),(7.5,1),(8.5,0.8)\}$ and the line determined the linear program (3).

Linear programming for longer OLED-lifetime

Organic Light Emitting Diodes (OLEDs) are considered as the display technology of the future and more and more commercial products are equipped with such displays as shown in Fig. 3. However, the cheapest OLED technology suffers from short lifetimes. We will show in this section how linear programming can be used to increase the lifetime of such displays.

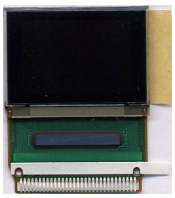

Fig. 3 Sample of a commercial OLED device with integrated driver chip

A (passive matrix) OLED display has a matrix structure consisting of n rows and m columns. At any crossover between a row and a column there is a vertical diode which works as a pixel. The image itself is given as an integral non-negative $n \times m$ matrix $(r_{ij}) \in [0,\ldots,\varrho]^{n\times m}$ representing its RGB values. Consider the contacts for the rows and columns as switches. For the time the switch of row i and column j is closed, an electrical current flows through the diode of pixel (i,j) and it shines. Hence, we can control the intensity of a pixel by the two quantities *electrical current* and *time*. The value r_{ij} determines the amount of time within the time frame in which the switches i and j have to be simultaneously closed. At a sufficient high frame rate e.g. 50 Hz, the perception by the eye is the average value of the light emitted by the pixel and one sees the image.

The traditional addressing scheme is row-by-row. This means that the switch for the first row is closed for a certain time while the switches for the columns are

closed for the necessary amount of time dictated by the entries r_{1j}, $j = 1,\dots,m$. Consequently the first row can be displayed in time $\max\{r_{1j} : j = 1,\dots,m\}$. Then the second row is displayed and so on. With this addressing scheme, the pixels are idle most of the time and then have to shine with very high intensity. This puts the diodes under stress and is a major cause of the short lifetime of the displays.

How can this lifetime problem be dealt with? The main idea is to save time, or equivalently to lower the maximum intensity, by displaying several rows at once.

Consider the schematic image on the left of Fig. 4. Let us compute the amount of time which is necessary to display the image with this addressing scheme. The maximum value of the entries in the first row is 238. This is the amount of time which is necessary to display the first row. After that the second row is displayed in time 237. In total the time which is required to display the image is $238+237+234+232+229 = 1170$ time units.

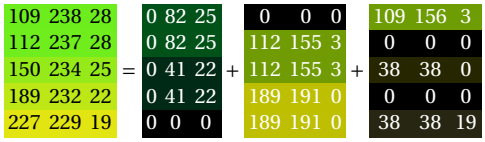

Fig. 4 An example decomposition

Now consider the decomposition of the image as the sum of the three images on the right of Fig. 4. In the first image, each odd row is equal to its even successor. This means that we can close the switches for rows 1 and 2 simultaneously, and these two equal rows are displayed in 82 time units. Rows 3 and 4 can also be displayed simultaneously which shows that the first image on the right can be displayed in $82+41$ time units. The second image on the right can be displayed in $155+191$ time units while the third image has to be displayed traditionally. In total all three images, and thus the original image on the left via this decomposition, can be displayed in $82+41+155+191+156+38+38 = 701$ time units. This means that we could reduce the necessary time via this decomposition by roughly 40%. We could equally display the image in the original 1170 time units but reduce the peak intensity, or equally the maximum electrical current through a diode by roughly 40%.

We now show how to model the time-optimal decomposition of an image as a linear program. To decompose R we need to find matrices $F^{(1)} = (f^{(1)}_{ij})$ and $F^{(2)} = (f^{(2)}_{ij})$ where $F^{(1)}$ represents the singleline part and $F^{(2)}$ the two doubleline parts. More precisely, the i-th row of matrix $F^{(2)}$ represents the doubleline covering rows i and $i+1$. Since the overlay (addition) of the subframes must be equal to the original image to get a valid decomposition of R, the matrices $F^{(1)}$ and $F^{(2)}$ must fulfill the constraint $f^{(1)}_{ij} + f^{(2)}_{i-1,j} + f^{(2)}_{ij} = r_{ij}$ for $i = 1,\dots,n$ and $j = 1,\dots,m$, where we now and in the following use the convention to simply omit terms with

indices running out of bounds. Since we cannot produce "negative" light we require also non-negativity of the variables $f_{ij}^{(\alpha)} \geqslant 0$. The goal is to find an integral decomposition that minimizes

$$\sum_{i=1}^{n} \max\{f_{ij}^{(1)} : 1 \leqslant j \leqslant m\} + \sum_{i=1}^{n-1} \max\{f_{ij}^{(2)} : 1 \leqslant j \leqslant m\} \ .$$

This problem can be formulated as a linear program by replacing the objective by $\sum_{i=1}^{n} u_i^{(1)} + \sum_{i=1}^{n-1} u_i^{(2)}$ and by adding the constraints $f_{ij}^{(\alpha)} \leqslant u_i^{(\alpha)}$. This yields

$$\begin{aligned} \min \ & \sum_{i=1}^{n} u_i^{(1)} + \sum_{i=1}^{n-1} u_i^{(2)} & & \\ \text{s.t.} \ & f_{ij}^{(1)} + f_{i-1,j}^{(2)} + f_{ij}^{(2)} = r_{ij} & \text{for all } i,j & \qquad (4) \\ & f_{ij}^{(\alpha)} \leqslant u_i^{(\alpha)} & \text{for all } i,j,\alpha & \end{aligned}$$

Note that the objective does not contain the f-variables. By decomposing images like this, the average lifetime of an OLED display can be increased by roughly 100%, see [1].

Exercises

1) A company produces and sells two different products. Our goal is to determine the number of units of each product they should produce during one month, assuming that there is an unlimited demand for the products, but there are some constraints on production capacity and budget.
   There are 20000 hours of machine time in the month. Producing one unit takes 3 hours of machine time for the first product and 4 hours for the second product. Material and other costs for producing one unit of the first product amount to 3CHF, while producing one unit of the second product costs 2CHF. The products are sold for 6CHF and 5CHF per unit, respectively. The available budget for production is 4000CHF initially. 25% of the income from selling the first product can be used immediately as additional budget for production, and so can 28% of the income from selling the second product.

   a. Formulate a linear program to maximize the profit subject to the described constraints.
   b. Solve the linear program graphically by drawing its set of feasible solutions and determining an optimal solution from the drawing.
   c. Suppose the company could modernize their production line to get an additional 2000 machine hours for the cost of 400CHF. Would this investment pay off?

2) A factory produces two different products. To create one unit of product 1, it needs one unit of raw material A and one unit of raw material B. To create one unit of product 2, it needs one units of raw material B and two units of raw material C. Raw material B needs preprocessing before it can be used, which takes one minute per unit. At most 20 hours of time is available per day for the preprocessing. Raw materials of capacity at most 1200 can be delivered to the factory per day. One unit of raw material A, B and C has size 4, 3 and 2 respectively.
   At most 130 units of the first and 100 units of the second product can be sold per day. The first product sells for 6 CHF per unit and the second one for 9 CHF per unit.
   Formulate the problem of maximizing turnover as a linear program in two variables and solve it.
3) Prove the following statement or give a counterexample: The set of optimal solutions of a linear program is always finite.
4) Let (5) be a linear program in inequality standard form, i.e.

$$\max\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n\} \tag{5}$$

   where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$.
   Prove that there is an equivalent linear program (6) of the form

$$\max\{\tilde{c}^T x \mid \tilde{A}x = \tilde{b}, x \geq 0, x \in \mathbb{R}^{\tilde{n}}\} \tag{6}$$

   where $\tilde{A} \in \mathbb{R}^{\tilde{m} \times \tilde{n}}$, $\tilde{b} \in \mathbb{R}^{\tilde{m}}$, and $\tilde{c} \in \mathbb{R}^{\tilde{n}}$ are such that every feasible point of (5) corresponds to a feasible point of (6) with the same objective function value and vice versa.
   Linear programs of the form in (6) are said to be in *equality standard form*.
5) Model the linear program (4) to decompose the EPFL logo with Zimpl. An incomplete model containing the encoding of the grayscale values of the logo can be found here here[2].
   Use an LP solver library of your choice to compute an optmal solution.
6) Provide an example of a convex and closed set $K \subseteq \mathbb{R}^2$ and a linear objective function $c^T x$ such that $\inf\{c^T x \colon x \in K\} > -\infty$ but there does not exist an $x^* \in K$ with $c^T x^* \leqslant c^T x$ for all $x \in K$.

References

1. F. Eisenbrand, A. Karrenbauer, and C. Xu. Algorithms for longer oled lifetime. In C. Demetrescu, editor, *6th International Workshop on Experimental Algorithms, WEA07*, volume 4525 of *Lecture Notes in Computer Science*, pages 338–351. Springer, 2007.
2. T. Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, 2004. ZIB-Report 04-58.

[2] http://disopt.epfl.ch/webdav/site/disopt/users/190205/public/logo_dec.zmpl

3. J. Matouek and B. Gärtner. *Understanding and Using Linear Programming (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.