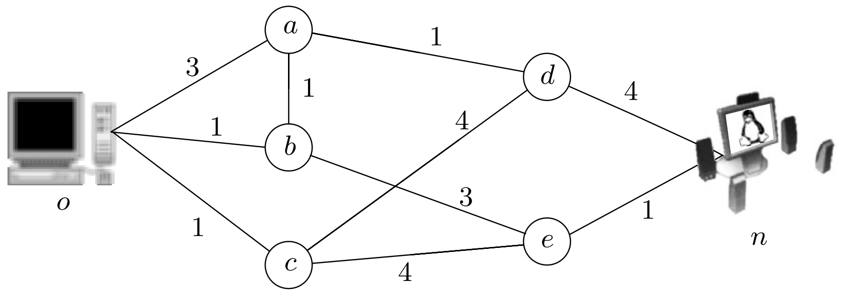
such problems here (many examples can be found in Chvátal's book cited in Chapter 9), and we will present problems in which the use of linear programming has different flavors.

2.2 Flow in a Network

An administrator of a computer network convinced his employer to purchase a new computer with an improved sound system. He wants to transfer his music collection from an old computer to the new one, using a local network. The network looks like this:



What is the maximum transfer rate from computer o (old) to computer n (new)? The numbers near each data link specify the maximum transfer rate of that link (in Mbit/s, say). We assume that each link can transfer data in either direction, but not in both directions simultaneously. So, for example, through the link ab one can *either* send data from a to b at any rate from 0 up to 1 Mbit/s, *or* send data from b to a at any rate from 0 to 1 Mbit/s.

The nodes $a, b, \ldots, e$ are not suitable for storing substantial amounts of data, and hence all data entering them has to be sent further immediately. From this we can already see that the maximum transfer rate cannot be used on all links simultaneously (consider node a, for example). Thus we have to find an appropriate value of the data flow for each link so that the total transfer rate from o to n is maximum.

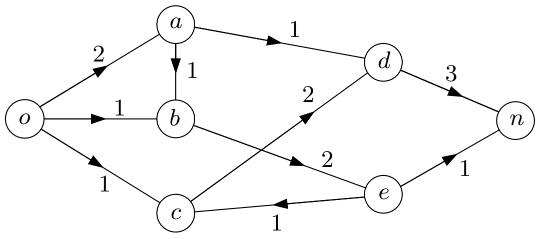
For every link in the network we introduce one variable. For example, x_{be} specifies the rate by which data is transfered from b to e. Here x_{be} can also be negative, which means that data flow in the opposite direction, from e to b. (And we thus do *not* introduce another variable x_{eb}, which would correspond to the transfer rate from e to b.) There are 10 variables: x_{oa}, x_{ob}, x_{oc}, x_{ab}, x_{ad}, x_{be}, x_{cd}, x_{ce}, x_{dn}, and x_{en}.

We set up the following linear program:

$$\begin{array}{ll}
\text{Maximize} & x_{oa} + x_{ob} + x_{oc} \\
\text{subject to} & -3 \le x_{oa} \le 3, \quad -1 \le x_{ob} \le 1, \quad -1 \le x_{oc} \le 1 \\
& -1 \le x_{ab} \le 1, \quad -1 \le x_{ad} \le 1, \quad -3 \le x_{be} \le 3 \\
& -4 \le x_{cd} \le 4, \quad -4 \le x_{ce} \le 4, \quad -4 \le x_{dn} \le 4 \\
& -1 \le x_{en} \le 1 \\
& \qquad x_{oa} = x_{ab} + x_{ad} \\
& x_{ob} + x_{ab} = x_{be} \\
& \qquad x_{oc} = x_{cd} + x_{ce} \\
& x_{ad} + x_{cd} = x_{dn} \\
& x_{be} + x_{ce} = x_{en}.
\end{array}$$

The objective function $x_{oa} + x_{ob} + x_{oc}$ expresses the total rate by which data is sent out from computer o. Since it is neither stored nor lost (hopefully) anywhere, it has to be received at n at the same rate. The next 10 constraints, $-3 \le x_{oa} \le 3$ through $-1 \le x_{en} \le 1$, restrict the transfer rates along the individual links. The remaining constraints say that whatever enters each of the nodes a through e has to leave immediately.

The optimal solution of this linear program is depicted below:



The number near each link is the transfer rate on that link, and the arrow determines the direction of the data flow. Note that between c and e data has to be sent in the direction from e to c, and hence $x_{ce} = -1$. The optimum value of the objective function is 4, and this is the desired maximum transfer rate.

In this example it is easy to see that the transfer rate cannot be larger, since the total capacity of all links connecting the computers o and a to the rest of the network equals 4. This is a special case of a remarkable theorem on maximum flow and minimum cut, which is usually discussed in courses on graph algorithms (see also Section 8.2).

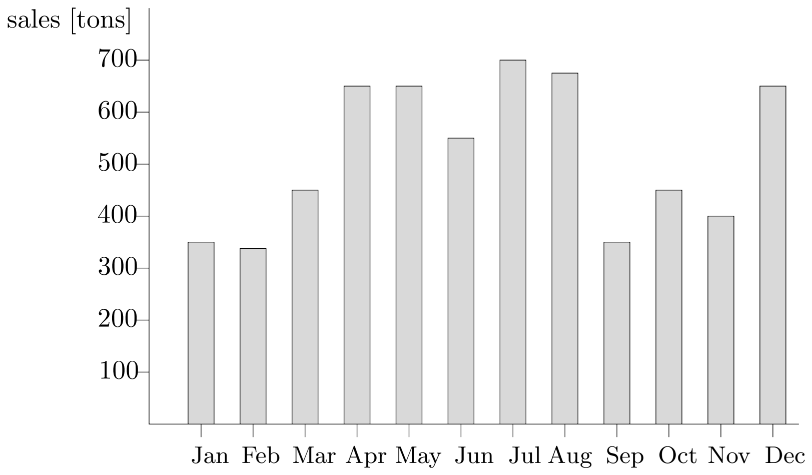
Our example of data flow in a network is small and simple. In practice, however, flows are considered in intricate networks, sometimes even with many source nodes and sink nodes. These can be electrical networks (current flows), road or railroad networks (cars or trains flow), telephone networks (voice or data signals flow), financial (money flows), and so on. There are also many less-obvious applications of network flows—for example, in image processing.

Historically, the network flow problem was first formulated by American military experts in search of efficient ways of disrupting the railway system of the Soviet block; see

> A. Schrijver: On the history of the transportation and maximum flow problems, *Math. Programming Ser. B* 91(2002) 437–445.

2.3 Ice Cream All Year Round

The next application of linear programming again concerns food (which should not be surprising, given the importance of food in life and the difficulties in optimizing sleep or love). The ice cream manufacturer Icicle Works Ltd.[2] needs to set up a production plan for the next year. Based on history, extensive surveys, and bird observations, the marketing department has come up with the following prediction of monthly sales of ice cream in the next year:



Now Icicle Works Ltd. needs to set up a production schedule to meet these demands.

A simple solution would be to produce "just in time," meaning that all the ice cream needed in month i is also produced in month i, $i = 1, 2, \ldots, 12$. However, this means that the produced amount would vary greatly from month to month, and a change in the produced amount has significant costs: Temporary workers have to be hired or laid off, machines have to be adjusted,

[2] Not to be confused with a rock group of the same name. The name comes from a nice science fiction story by Frederik Pohl.

and so on. So it would be better to spread the production more evenly over the year: In months with low demand, the idle capacities of the factory could be used to build up a stock of ice cream for the months with high demand.

So another simple solution might be a completely "flat" production schedule, with the same amount produced every month. Some thought reveals that such a schedule need not be feasible if we want to end up with zero surplus at the end of the year. But even if it is feasible, it need not be ideal either, since storing ice cream incurs a nontrivial cost. It seems likely that the best production schedule should be somewhere between these two extremes (production following demand and constant production). We want a compromise minimizing the total cost resulting both from changes in production and from storage of surpluses.

To formalize this problem, let us denote the demand in month i by $d_i \geq 0$ (in tons). Then we introduce a nonnegative variable x_i for the production in month i and another nonnegative variable s_i for the total surplus in store at the end of month i. To meet the demand in month i, we may use the production in month i and the surplus at the end of month $i-1$:

$$x_i + s_{i-1} \geq d_i \quad \text{for } i = 1, 2, \ldots, 12.$$

The quantity $x_i + s_{i-1} - d_i$ is exactly the surplus after month i, and thus we have

$$x_i + s_{i-1} - s_i = d_i \quad \text{for } i = 1, 2, \ldots, 12.$$

Assuming that initially there is no surplus, we set $s_0 = 0$ (if we took the production history into account, s_0 would be the surplus at the end of the previous year). We also set $s_{12} = 0$, unless we want to plan for another year.

Among all nonnegative solutions to these equations, we are looking for one that minimizes the total cost. Let us assume that changing the production by 1 ton from month $i-1$ to month i costs €50, and that storage facilities for 1 ton of ice cream cost €20 per month. Then the total cost is expressed by the function

$$50 \sum_{i=1}^{12} |x_i - x_{i-1}| + 20 \sum_{i=1}^{12} s_i,$$

where we set $x_0 = 0$ (again, history can easily be taken into account).

Unfortunately, this cost function is not linear. Fortunately, there is a simple but important trick that allows us to make it linear, at the price of introducing extra variables.

The change in production is either an increase or a decrease. Let us introduce a nonnegative variable y_i for the increase from month $i-1$ to month i, and a nonnegative variable z_i for the decrease. Then

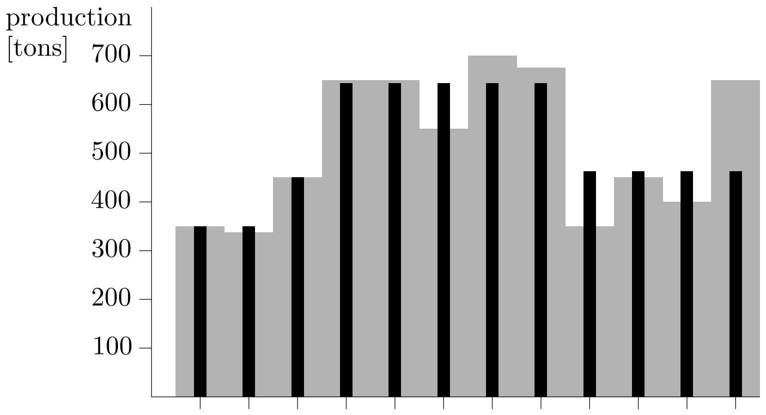
$$x_i - x_{i-1} = y_i - z_i \text{ and } |x_i - x_{i-1}| = y_i + z_i.$$

A production schedule of minimum total cost is given by an optimal solution of the following linear program:

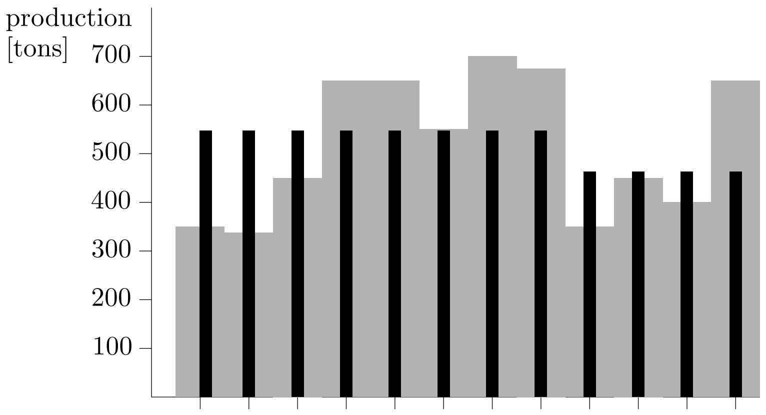
$$\begin{array}{ll}\text{Minimize} & 50\sum_{i=1}^{12} y_i + 50\sum_{i=1}^{12} z_i + 20\sum_{i=1}^{12} s_i \\ \text{subject to} & x_i + s_{i-1} - s_i = d_i \text{ for } i = 1,2,\ldots,12 \\ & x_i - x_{i-1} = y_i - z_i \text{ for } i = 1,2,\ldots,12 \\ & x_0 = 0 \\ & s_0 = 0 \\ & s_{12} = 0 \\ & x_i, s_i, y_i, z_i \geq 0 \text{ for } i = 1,2,\ldots,12.\end{array}$$

To see that an optimal solution $(\mathbf{s}^*, \mathbf{y}^*, \mathbf{z}^*)$ of this linear program indeed defines a schedule, we need to note that one of y_i^* and z_i^* has to be zero for all i, for otherwise, we could decrease both and obtain a better solution. This means that $y_i^* + z_i^*$ indeed equals the change in production from month $i-1$ to month i, as required.

In the Icicle Works example above, this linear program yields the following production schedule (shown with black bars; the gray background graph represents the demands).



Below is the schedule we would get with zero storage costs (that is, after replacing the "20" by "0" in the above linear program).

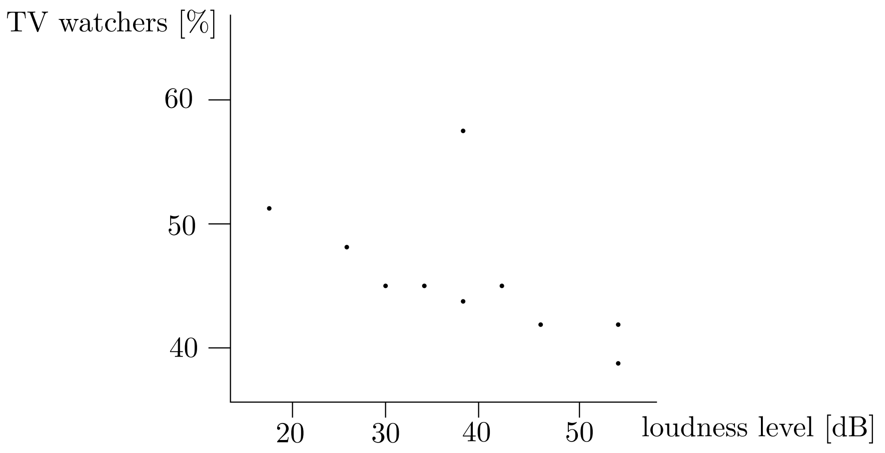


The pattern of this example is quite general, and many problems of optimal control can be solved via linear programming in a similar manner. A neat example is "Moon Rocket Landing," a once-popular game for programmable calculators (probably not sophisticated enough to survive in today's competition). A lunar module with limited fuel supply is descending vertically to the lunar surface under the influence of gravitation, and at chosen time intervals it can flash its rockets to slow down the descent (or even to start flying upward). The goal is to land on the surface with (almost) zero speed before exhausting all of the fuel. The reader is invited to formulate an appropriate linear program for determining the minimum amount of fuel necessary for landing, given the appropriate input data. For the linear programming formulation, we have to discretize time first (in the game this was done anyway), but with short enough time steps this doesn't make a difference in practice.

Let us remark that this particular problem can be solved analytically, with some calculus (or even mathematical control theory). But in even slightly more complicated situations, an analytic solution is out of reach.

2.4 Fitting a Line

The loudness level of nightingale singing was measured every evening for a number of days in a row, and the percentage of people watching the principal TV news was surveyed by questionnaires. The following diagram plots the measured values by points in the plane:



The simplest dependencies are linear, and many dependencies can be well approximated by a linear function. We thus want to find a line that best fits the measured points. (Readers feeling that this example is not sufficiently realistic can recall some measurements in physics labs, where the measured quantities should actually obey an exact linear dependence.)