Linnæus University
Optimization 2MA404/2MA918
Björn Lindenberg
Jonas Nordqvist
jonas.nordqvist@lnu.se

Computer Laboration I: 2MA404/2MA918

Instructions for hand-in: Submit a **SINGLE** pdf-file. (use LaTeXif you are used to it).

Start with a heading and your name. Continue with subheadings for the exercises. For each exercise the following needs to be handed in:

1: All figures produced, and your code. Answers to all questions.

2: A table/figure illustrating your experiments and answers to all questions.

3: Answers to all questions.

Please present your results with coherent sentences. Save the figures in a suitable format, so that they can be inserted in your Word- or TeX-document. Please use a different type face or color for code, so that it is easy to separate text and copy-pasted code.

Exercise 1: Warm-up LP-problem

A company produces two types of television sets, a cheap type (A) and an expensive type (B). The company has a profit of 700 per unit for type A and 1000 per unit for type B. There are three stages in the production process. In stage I, 3 hours are needed per unit of type A and 5 hours per unit of type B. The total number of available working hours in stage I is 3900. In step II, 1 hour is worked on each device of type A and for 3 hours on each device of type B. The total working time available is 2100 hours. In step III, 2 working hours are needed for both types, and 2200 working hours are available. How many TV sets of each type should the company produce to maximize its profit?

As a first step, we want to create matrices $A, \mathbf{b}$ and $\mathbf{c}$ such that the problem is formulated in terms of

$$\begin{aligned} \max_{c} \quad & \mathbf{c}^T\mathbf{x} = z \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned} \tag{1}$$

i) Define the model with decision variables x_1, x_2 and create $A, \mathbf{b}$ and $\mathbf{c}$ using `numpy` in Python.

ii) The feasible region is a convex set restricted by the x_1- and x_2-axis and the lines

$$a_{i,1}x_1 + a_{i,2}x_2 = b_i,$$

for $i = 1, 2, 3$. We want to plot these lines to get a good overview of the problem. Create a `numpy`-array using `linspace` that goes between 0 and 1200 with a suitable number of points, e.g.,

```
x = np.linspace(0, 1200, 100).
```

This now represents some x_1-values.

Reformulate each line equation as $y_i := \frac{1}{a_{i,2}}(b_i - a_{i,1}x)$ for $i = 1, 2, 3$ and compute the corresponding points of x_2-values using `x`. Plot all lines in the same figure by importing `matplotlib.pyplot` as `plt` in your script and then use `plt.plot` for each pair (x, y_i). Be sure to set a legend by supplying `plt.legend` with a list of three strings to clearly see which constraint each line represents (the order must follow the order of calls to `plt.plot`). Also set appropriate y-limits using

```
plt.ylim(0, 1200)
```

and call `plt.show()` in the end to visualize your figure.

iii) Study the figure produced in (ii). There are five vertices that enclose the feasible region. Each of these will be the intersection of two lines (including axis lines). Graphically determine their values, i.e., see which two constraints are satisfied with equality for each vertex. Store the vertices in a 5 by 2 `numpy`-array.

The feasible region X in our case is precisely the convex hull of the five vertices (smallest convex set containing them). So add an import of `ConvexHull` and `convex_hull_plot_2d` from `scipy.spatial` in your script. Create a hull and plot it all in one go by

```
convex_hull_plot_2d(ConvexHull(vertices)),
```

where `vertices` are your `numpy`-array of points. Adjust y-values with `plt.ylim(0, 800)`. Note that this will always begin a new figure and we will use this in the next step so delay the call to `plt.show()` a little bit (or move it down later).

iv) Let's also plot some level curves for $z = \mathbf{c}^T\mathbf{x}$ in the convex hull plot. Using the previous `x`-array for x_1-values, figure out what the corresponding x_2-values must be if $z = C$ for

$$C = 1 \cdot 10^5, 2 \cdot 10^5, \ldots, 8 \cdot 10^5.$$

If the x_2-values for a specific $C := k \cdot 10^5$ is $y(k)$, then plot all curves with a suitable `for`-loop indexed by k and using

`plt.plot(x, y(k), 'r', alpha=k/8)`.

This will now plot eight level curves of z on top of your hull with an increasing red color gradient for larger values of z. Determine from your plot in which vertex the maximum is obtained.

v) Verify by checking the value of z in all extreme points. This can be done by looping over each vertex `v` in your collection and using `c @ v` (`numpy` matrix multiplication).

vi) Now that we know the ground truth we will instead use `scipy` to find the maximum for us. Import `linprog` and `OptimizeResult` from `scipy.optimize`. Have a good look at the documentation of both[1] [2] (can also be found in their respective definitions). You will need to provide arguments to `linprog` in precisely the right way and then know how to interpret the result in the returned `OptimizeResult` (which is a dictionary). Recall that $\max z$ is equivalent to $\min(-z)$. Find and extract the maximum using your defined matrices $A, \mathbf{b}$ and $\mathbf{c}$.

vii) Setup the problem in *standard form* (using equality constraints with slack variables). Again make sure that you have a good grasp on the usage of `linprog`. Solve it and verify that the optimal solution stays the same.

Exercise 2: Large LP-problems

In this exercise we will investigate the time that it takes to solve a larger LP-problem.

In order to create a random large system which has a bounded, non-empty feasible region we have to select A, $\mathbf{b}$ and $\mathbf{c}$ carefully. Note also the imports for later.

```
import numpy as np
from numpy.random import rand, randn
from scipy.optimize import linprog, OptimizeResult
from time import perf_counter

# Example of a problem instance
n, m = 10, 10 # Freely changed
A = np.concatenate([rand(m, n) + 1, np.eye(m)], axis=-1)
b = np.ones(m)
c = np.concatenate([randn(n), np.zeros(m)])
```

Figure 1: Code for generating A, $\mathbf{b}$, and $\mathbf{c}$ for an LP-problem with a non-empty, bounded feasible region. Note copy-paste works poorly, rather write the code by hand.

i) By formulating the matrix A and the vectors $\mathbf{b}$, $\mathbf{c}$ in the above manner we guarantee a solution. Why?

ii) We warm up by solving the example instance using the simplex method by supplying `method = "simplex"` to `linprog`. Further, we can estimate the elapsed time for the procedure in milliseconds using `perf_counter()`.

[1] https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.linprog.html
[2] https://docs.scipy.org/doc/scipy-1.8.1/reference/generated/scipy.optimize.OptimizeResult.html

```
# Extract fractional seconds (high resolution)
start_time = perf_counter()
# Optimize with 'simplex'
result = linprog(-c,
                 A_eq=A,
                 b_eq=b,
                 method='simplex',
                 options={'maxiter': 5000})
# Est. elapsed time in milliseconds
elapsed_time = 1000*(perf_counter() - start_time)
```

Running a few times shows that it takes roughly 3 ms on my machine. How about your machine?

iii) Keep $m = n$ and steadily increase the number of variables using the `'simplex'` method. For each value, perform the experiment five times by re-sampling and then present the average elapsed time. On my current machine the average time exceeds 1 second at $m = n = 170$. How high must you go for the average time to exceed 1 second (1000 ms)?

iv) Now we will change the method to a sophisticated and state-of-the-art algorithm (with its roots in simplex). Supply the default LP-method *HiGHS* to `linprog`, either by removing the `method` keyword or passing `method = 'highs'`. On my current machine the average time exceeds 1 second at $m = n = 2800$. How high must you now go for the 1 second average?

Exercise 3: Sensitivity analysis

In this exercise we will perform a sensitivity analysis of the LP-problem in Exercise 1.

i) Formulate and solve the dual problem. Verify that the solutions are the same.

ii) Compute the shadow price of the various constraints.

iii) Suppose that you are granted 100 extra working hours in either stage I, II or III. Where would you invest this time, and where should you not invest it?

iv) Suppose that the company investigates the possibility to increase the price of the tv of type (B). How much must the price increase for the optimal solution to change?

v) A team of marketing people and some technicians have considered a new kind of tv to produce. Suppose that this new type of tv is called type (C). This is a premium tv and will return a profit of 1350. However, it takes some time to construct and the times to produce it are $(7, 4, 2)$ for each separate stage. Add this type of tv to the problem, and find the reduced cost for the type (C) tv. Consider the value of the reduced cost. Should the tv be produced or not? If so, re-run the optimization to get a new optimum.

vi) The company's new quality director have decided to add a new stage in the production for quality inspection. She claims that it'll take 0.5 hours to inspect the type (A) tv, and 0.75 hours to inspect the more advanced tv of type (B). One upside with producing tv of type (C) is the advanced production which involves quality inspection already in the production, so the inspection time is only 6 minutes. The CEO do not want to decrease production and is willing to pay for the quality inspections. How many hours must the company add to this working line, not to interfer with the current production amounts?