Exercise Sheet 5
LP, ILP, MILP & DP

C1 – Linear and Dynamic Programming

Exercise E5.1 *(Gurobi)*
Some of the following exercises make use of the MILP solver *Gurobi Optimizer*. You can download the program from `gurobi.com` after creating a free 'Academic' account.

To install the licence, put the following in a text file called `gurobi.lic` and save in your home directory (e.g. `~/gurobi.lic` on Linux / macOS or `C:\Users\Astronaut\gurobi.lic` on Windows):

```
TOKENSERVER=lizenzserver.hiz-saarland.de
PORT=46325
```

Caution: You need to be connected to the university network to use the Gurobi licence server, either directly or using a VPN connection at home.

Exercise E5.2 *(Knaprocket)*
Suppose you are planning the next rocket launch. Your company builds heavy-lift launch vehicles (HLLV) that can lift payloads of up to $\hat{w}$ kg altogether into orbit. Furthermore, you have contracts with n different satellite manufacturers that build one satellite of weight w_i kg each and are willing to pay you $\$c_i$ for every launched satellite. Your problem of interest is to figure out how to load the rocket with a combination of satellites of the specified manufacturers that yields the greatest total revenue.

(a) Model this problem as an integer linear program.

(b) The first approach is unrealistic in the sense that there is no restriction on the number of payloads. Now, consider the case when your launch vehicle has m payload adapters, i.e. can launch up to m satellites. How do you have to modify your ILP? Is this problem easier to solve than without the capacity restriction?

Exercise E5.3
Consider the following optimisation problem in $\mathbb{R}^2$.

$$\begin{array}{lrcrcr}
\text{minimise} & 2x_1 & + & x_2 & & \\
\text{subject to} & x_1 & + & x_2 & \geq & -5 \\
& -2x_1 & + & 2x_2 & \leq & 8 \\
& -16x_1 & + & 4x_2 & \geq & -5 \\
& & & x_1 & \leq & 0 \\
& x_1, x_2 \in \mathbb{Z} & & & &
\end{array}$$

(a) Draw the polyhedron.

(b) Find the optimal fractional *and* integer solution graphically.

Exercise E5.4 *(Propositional Logic)*
You are given a propositional formula φ consisting of the two constants top $\top$ and bottom $\bot$ and the connectives negation $\neg$, disjunction $\vee$, conjunction $\wedge$, and implication $\rightarrow$.

(a) Model the problem of determining the truth value of φ as an LP. Use one variable for each constant as well as one variable for each subexpression $\psi \star \chi$ where $\star \in \{\neg, \vee, \wedge, \rightarrow\}$. Describe which constraints you need to add for each type of connectives.

(b) Argue why we don't need any integrality constraints on the decision variables by showing that the variables always take integer values, i.e. 0 or 1.

Exercise E5.5 *(Absolute Values)*
Consider the problem

$$\begin{array}{ll} \text{minimise} & 3x_1 + 7\,|x_2 - 21| \\ \text{subject to} & |x_1 + 4| + |x_2| \le 9 \end{array}$$

and reformulate it as an equivalent linear programming problem (i.e. without the absolute values).

Exercise E5.6 *(Diet Problem)*
Suppose you are an astronaut on the ISS. In the kitchen, there are n different foods available that contain m different nutrients. We are given the nutritional content a_{ij} of nutrient $i \in \{1, \dots, m\}$ of a unit of food $j \in \{1, \dots, n\}$. Each food j costs $\$c_j$. In order to stay healthy, you have to eat at least b_i units of nutrient i per day. Your task is to synthesise the ideal food at minimal costs that satisfies the minimal nutrient requirements.

(a) Formulate the problem of finding the ideal diet as an integer linear program.

(b) Use Gurobi to calculate the ideal diet using the following data. How much does the diet cost?

Food	Protein	Carbohydrates	Fats	Cost
Pizza	11	33	10	5
Fish	22	0	12	8
Eggplant	1	6	0	1
Coke	0	11	0	2
Demand	50	310	70	

(c) Try also to formulate the general problem with n foods and m nutrients in Gurobi.

Hint: You can use Python List Comprehensions.

Exercise E5.7 *(Shortest Paths)*
Let $G = (V, A)$ be a directed graph, $s, t \in V$ two nodes, and c_{vw} the costs for each edge (v, w). Formulate the problem of finding the shortest path, i.e. the path with minimal costs, from s to t as a linear program.

Exercise E5.8 *(Levenshtein)*
The *Levenshtein distance* is a string metric for measuring the difference between two words. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions, or substitutions) required to change one word into the other.

> **Example:** For example, the Levenshtein distance between 'kitten' and 'sitting' is 3, since the following 3 edits change one into the other, and there is no way to do it with fewer than 3 edits:
>
> 1. **k**itten → **s**itten (substitution of 's' for 'k')
> 2. sitt**e**n → sitt**i**n (substitution of 'i' for 'e')
> 3. sittin → sittin**g** (insertion of 'g' at the end)

Formally, the Levenshtein distance between two strings a and b (of length $|a|$ and $|b|$ respectively) is given by $\operatorname{lev}(a, b)$, where:

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0 \\ |b| & \text{if } |a| = 0 \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) & \text{if } \operatorname{head}(a) = \operatorname{head}(b) \\ 1 + \min \left\{ \begin{array}{l} \operatorname{lev}(\operatorname{tail}(a), b) \\ \operatorname{lev}(a, \operatorname{tail}(b)) \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) \end{array} \right\} & \text{otherwise} \end{cases}$$

Looking closely at this recursive definition, we see that the problem has an optimal substructure as well as overlapping subproblems (i.e. the same subproblems are repeatedly computed). Therefore, this problem is well-suited to be solved using dynamic programming.

Design an algorithm for the Levenshtein distance $\operatorname{lev}(a, b)$ using dynamic programming. You may write pseudocode or use a real programming language like Python. Explain how you design your DP table to memorize already computed subproblem solutions.