

Exercises for Mathematical Programming

186.835 VU 3.0 – SS 2021

Fabian Holzberger 11921655

Exercise 1 Network Design (1 point)

We are given a directed graph $G = (V, A)$ and a value $b_i \in \mathbb{R}$ for each node $i \in V$ which denotes a demand ($b_i < 0$) or a supply ($b_i > 0$), such that $\sum_{i \in V} b_i = 0$. There are two types of costs: transportation costs c_{ij} of shipping one unit from node i to node j , and building costs of establishing a direct link (i, j) from node i to j . A link on arc (i, j) can be built (i) with costs d_{ij}^1 and capacity u_{ij}^1 , or (ii) with costs d_{ij}^2 and capacity u_{ij}^2 . Assume that $d_{ij}^1 < d_{ij}^2$ and $u_{ij}^1 < u_{ij}^2$ and that at most one option can be chosen on each arc. A network has to be built that satisfies all demands and minimizes the total building and transportation costs. Formulate the problem as a (mixed) integer linear program.

Answer: Assume we model flow from node i to node j over an arc of type 1 by the variable $x_{ij}^1 \in \mathbb{R}$ and of type 2 respectively by $x_{ij}^2 \in \mathbb{R}$. Further let $y_{ij}^1, y_{ij}^2 \in \{0, 1\}$ be boolean variables that express if we have an arc of type 1 or 2 between (i, j) or not. Then the objective function is given by:

$$g = \sum_{\substack{i,j \\ (i,j) \in A}} \sum_{\alpha \in \{1,2\}} [x_{ij}^\alpha c_{ij} + d_{ij}^\alpha y_{ij}^\alpha] \quad (1)$$

If we have an arc (i, j) it shall be either type 1 or 2, which we model by the constraint $0 \leq y_{ij}^1 + y_{ij}^2 \leq 1 \quad \forall i, j ((i, j) \in A)$. The variables y_{ij}^α and x_{ij}^α are coupled by the constraints $0 \leq x_{ij}^\alpha \leq y_{ij}^\alpha u_{ij}^\alpha \quad \forall i, j ((i, j) \in A)$ and $\forall \alpha \in \{1, 2\}$. This ensures that the flow over a connection of type 1 or 2 is only nonzero if the connection exists and that it will not exceed the capacity of the respective arc. Lastly we ensure that the demand of every node is fulfilled by: $b_j + \sum_i \sum_\alpha (x_{ij}^\alpha - x_{ji}^\alpha) \geq 0 \quad \forall j \in V$, which states that the sum of outflows and inflows at node j must satisfy the demand. Therefore, we get the following MILP:

$$\text{find } x_{ij}^\alpha \in \mathbb{R} \text{ and } y_{ij}^\alpha \in \{0, 1\} \quad \forall i, j ((i, j) \in A) \text{ and } \forall \alpha \in \{1, 2\} : \quad (2)$$

$$\min \sum_{\substack{i,j \\ (i,j) \in A}} \sum_{\alpha \in \{1,2\}} [x_{ij}^\alpha c_{ij} + d_{ij}^\alpha y_{ij}^\alpha] \quad (3)$$

$$0 \leq y_{ij}^1 + y_{ij}^2 \leq 1 \quad \forall i, j ((i, j) \in A) \quad (4)$$

$$0 \leq x_{ij}^\alpha \leq y_{ij}^\alpha u_{ij}^\alpha \quad \forall i, j ((i, j) \in A) \text{ and } \alpha \in \{1, 2\} \quad (5)$$

$$b_j + \sum_{i \in V} \sum_{\alpha \in \{1,2\}} (x_{ij}^\alpha - x_{ji}^\alpha) \geq 0 \quad \forall j \in V \quad (6)$$

Exercise 2 Scheduling (1 point)

A factory consists of m machines M_1, \dots, M_m . Each job $j = 1, \dots, n$ needs to be processed on each machine in the specific order $M_{j(1)}, \dots, M_{j(m)}$ (permutation of the machines). Machine M_i takes time p_{ij} to process job j . A machine can only process one job at a time, and once a job is started on any machine, it must be processed to completion. The objective is to minimize the **sum of the completion times** of all the jobs. The completion time of job j is the time when the last subtask on machine $M_{j(m)}$ is finished. Formulate the problem as a (mixed) integer linear program.

Answer: Let M, N the indexsets of machines and jobs. Assume the variable $x_{kj} \in \mathbb{R}$ defines the start time of a job j on machine M_k . Jobs can't overlap on a machine, and we want that the condition $(x_{kj} - x_{ki} \geq p_{ki}) \vee (x_{ki} - x_{kj} \geq p_{kj})$ holds. One can express this condition with aid of a new variable y_{kji} , that shall be 1 if job j is starting before i on machine M_k and 0 else, as done in [1]. Therefore let $y_{kji} \in \mathbb{N}$ and $0 \leq y_{kji} \leq 1$. Next define the number $T_\infty := \sum_{k,i} p_{ki} \geq \sup_{i,j} |x_{li} - x_{lj}| \quad \forall l \in M$. Then we have the conditions:

$$(T_\infty + p_{ki})(1 - y_{kji}) + (x_{kj} - x_{ki}) \geq p_{ki} \quad \forall i, j \in N \text{ and } \forall k \in M \quad (7)$$

$$(T_\infty + p_{kj})y_{kji} + (x_{ki} - x_{kj}) \geq p_{kj} \quad \forall i, j \in N \text{ and } \forall k \in M \quad (8)$$

Note that the conditions imply that if $(x_{ki} - x_{kj}) > 0$ and (7) holds then $y_{kji} = 0$ and on the other side if $(x_{ki} - x_{kj}) \leq 0$ and (8) hold then $y_{kji} = 1$, as required. Further, we see that the two conditions ensure that start times can not overlap. To ensure that a job j is executed in its right order $M_{j(1)}, \dots, M_{j(m)}$ we introduce the variable z_{hkj} that is 1 if the

job j needs to be executed on machine M_h before it can be executed on M_k and 0 otherwise. We can precompute the value of $z_{h kj}$ and therefore add the last condition:

$$z_{h kj}(x_{hj} + p_{hj}) \leq x_{kj} \quad \forall j \in N \text{ and } \forall k, h \in M \quad (9)$$

For the objective function we note that the index of the last machine that job j needs to be processed on is given by $M_{j(m)}$ and therefore the objective is : $\sum_{j \in N} (x_{M_{j(m)}j} + p_{M_{j(m)}j})$. The full MILP is therefore:

$$\text{find } x_{kj} \in \mathbb{R} \text{ and } y_{kj} \in \{0, 1\} \quad \forall j \in N \text{ and } \forall k \in M : \quad (10)$$

$$\min \sum_{j \in N} (x_{M_{j(m)}j} + p_{M_{j(m)}j}) \quad (11)$$

$$(T_\infty + p_{ki})(1 - y_{kji}) + (x_{kj} - x_{ki}) \geq p_{ki} \quad \forall i, j \in N \text{ and } \forall k \in M \quad (12)$$

$$(T_\infty + p_{kj})y_{kji} + (x_{ki} - x_{kj}) \geq p_{kj} \quad \forall i, j \in N \text{ and } \forall k \in M \quad (13)$$

$$z_{h kj}(x_{hj} + p_{hj}) \leq x_{kj} \quad \forall j \in N \text{ and } \forall h, k \in M \quad (14)$$

$$x_{kj} \geq 0 \quad \forall k \in M \text{ and } \forall j \in N \quad (15)$$

$$\text{where } z_{h kj} := \begin{cases} 1 & \text{if } M_j^{-1}(h) < M_j^{-1}(k) \\ 0 & \text{else} \end{cases} \quad (16)$$

Exercise 3 Sports League (3 points)

The season in a sports league with n teams has started. Each team plays against each other team at home and away, so each team plays exactly $2(n-1)$ games. In case of a win, a team receives three points, in case of a draw one point, and in case of a loss zero points. After the season the k worst teams with respect to the total number of achieved points get relegated. The task is now to determine the minimum number of points a team has to achieve to have a guarantee that it is not relegated in any season outcome.

- Formulate the problem as a (mixed) integer linear program. (1 point)
- Solve the problem for $n = 18$ and $k = 3$ with an MILP solver (e.g., CPLEX, Gurobi, Excel Solver). (2 points)

Answer: Let us denote by $N := \{0, 1, \dots, n-1\}$ the index-set of teams. To model the outcome of the games we create three variables indexed by the games $x_{ij}^w, x_{ij}^d, x_{ij}^l \in \{0, 1\}$ where the first is one if team i wins against j at home and 0 otherwise, the second one is 1 if the game is a draw and the third 1 if i loses and therefore j wins. Note that we only create variables for $i \neq j$ and therefore can model a game at home for team i by the index ij and a game away by ji . A team can either win, loose or have a draw with another team. Therefore, we have the constraint:

$$x_{ij}^w + x_{ij}^d + x_{ij}^l = 1 \quad \forall i, j \in N (i \neq j) \quad (17)$$

The points of a team can be described by the points earned in home games and away games:

$$p_t = \sum_{j, j \neq t} (3x_{tj}^w + x_{tj}^d + 3x_{jt}^l + x_{jt}^d) \quad (18)$$

Note that for an away game of the team t we have the indices jt such that x_{jt}^w is 1 if j wins but x_{jt}^l if t wins, which is the reason we include it above. Next we want to order the points from $t = 0$ with the lowest points to $t = n$ that achieves the highest points. Therefore let $p_t \leq p_{t+1} \quad \forall t \in N \setminus \{n-1\}$. The objective is then to maximize p_k such that we then know that if a team has $k+1$ points it is surely not relegated in any season outcome. The MILP taken from [2] is therefore:

$$\text{find } x_{ij}^w, x_{ij}^d, x_{ij}^l \in \{0, 1\} \quad \forall i, j \in N : \quad (19)$$

$$\max p_k = \max \left[\sum_{j, j \neq k} (3x_{kj}^w + x_{kj}^d + 3x_{jk}^l + x_{jk}^d) \right] \quad \text{for } k \in N \quad (20)$$

$$x_{ij}^w + x_{ij}^d + x_{ij}^l = 1 \quad \forall i, j \in N (i \neq j) \quad (21)$$

$$\sum_{j, j \neq t} (3x_{tj}^w + x_{tj}^d + 3x_{jt}^l + x_{jt}^d) \leq \sum_{j, j \neq t+1} (3x_{t+1j}^w + x_{t+1j}^d + 3x_{jt+1}^l + x_{jt+1}^d) \quad \forall t \in N \setminus \{n-1\} \quad (22)$$

$$(23)$$

We want to solve the the MILP for $n = 18$ and $k = 3$ by the Python MIP package ([link](#)).

Listing 1:

```

1 import pip
2 import numpy as np
3 from mip import Model, xsum, maximize, BINARY
4 pip.main(['install', 'mip'])
5
6 nt = 18
7 I = range(nt) # from 0 to 17
8 k = 2 # maximize for team k, we start at 0!!
9
10 m = Model("Sports League")
11
12 xw = []
13 xd = []
14 xl = []
15
16 for i in I:
17     xw.append([])
18     xd.append([])
19     xl.append([])
20     for j in I:
21         if i!=j:
22             xw[i].append(m.add_var(var_type=BINARY))
23             xd[i].append(m.add_var(var_type=BINARY))
24             xl[i].append(m.add_var(var_type=BINARY))
25             m += (xw[i][j] + xd[i][j] + xl[i][j]) == 1
26         else:
27             xw[i].append([])
28             xd[i].append([])
29             xl[i].append([])
30
31 for t in range(nt-1):
32     m += xsum( 3*xl[i][t] + xd[i][t] +
33               3*xw[t][i] + xd[t][i] for i in I if i!=t) <= \
34         xsum( 3*xl[i][t+1] + xd[i][t+1] +
35               3*xw[t+1][i] + xd[t+1][i] for i in I if i!=(t+1))
36
37 m.objective = maximize(xsum( 3*xl[i][k] + xd[i][k] +
38                             3*xw[k][i] + xd[k][i] for i in I if i!=k))
39 m.optimize()
40
41 pk = np.sum([3* int(xl[i][k].x) + int(xd[i][k].x) +
42              3* int(xw[k][i].x) + int(xd[k][i].x) for i in I if i!=k])
43
44 print("Team k={} has {} points and will still relegate.".format(k,pk)+\
45       "Therefore a team is safe if it has at least {} points".format(pk+1))

```

The result is that the team 0 and 1 loose all games except when they play against each other they have a draw. Therefore, team 0 and 1 have 2 points and all other teams 57 points, including team 2. Therefore, a team will never relegate if it reaches 58 points.

References

- [1] A. S. Manne, “On the job-shop scheduling problem,” *Operations Research*, vol. 8, no. 2, pp. 219–223, 1960.
- [2] C. Raack, A. Raymond, A. Werner, and T. Schlechte, “Integer programming and sports rankings,” 2013.