MIT 15.083: Mixed-Integer Programming **Spring 2021**

Problem Set 4

Student Name: Léonard Boussioux

Problem 1

The original problem is:

$$\begin{aligned}
&\min \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum_{s\in\mathcal{S}} p_s \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} f_{ijk} y^s_{ijk} \\
&\text{s.t.} \\
&\textstyle\sum_{j:(i,j)\in\mathcal{A}} y^s_{ijk} - \sum_{j:(j,i)\in\mathcal{A}} y^s_{jik} = \begin{cases} d^s_k & i = O_k \\ 0 & i \neq O_k, D_k \\ -d^s_k & i = D_k \end{cases} \quad \forall i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S} \\
&\sum_{k\in\mathcal{K}} y^s_{ijk} \leq u_{ij}x_{ij}, \forall (i,j) \in \mathcal{A}, s \in \mathcal{S} \\
&x_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{A} \\
&y^s_{ijk} \in \{0,1\}, \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, s \in \mathcal{S}
\end{aligned} \tag{4.1}$$

We write the subproblem s:

$$\begin{aligned}
&\min \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} f_{ijk} y^s_{ijk} \\
&\text{s.t.} \\
&\textstyle\sum_{j:(i,j)\in\mathcal{A}} y^s_{ijk} - \sum_{j:(j,i)\in\mathcal{A}} y^s_{jik} = \begin{cases} d^s_k & i = O_k \\ 0 & i \neq O_k, D_k \\ -d^s_k & i = D_k \end{cases} \quad \forall i \in \mathcal{N}, k \in \mathcal{K} \\
&\sum_{k\in\mathcal{K}} y^s_{ijk} \leq u_{ij}x_{ij}, \forall (i,j) \in \mathcal{A} \\
&y^s_{ijk} \in \{0,1\}, \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}
\end{aligned} \tag{4.2}$$

The corresponding dual subproblem s writes:

$$\begin{aligned}
\max \quad & \sum_{k\in\mathcal{K}} d^s_k(\lambda^s_{O_k,k} - \lambda^s_{D_k,k}) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu^s_{ij} \\
\text{s.t.} \quad & \lambda^s_{i,k} - \lambda^s_{j,k} + \mu^s_{ij} \leq f_{ijk}, \forall (ij) \in \mathcal{A}, k \in \mathcal{K} \\
& \lambda^s_{ik} \in \mathbb{R}, \forall i \in \mathcal{N}, k \in \mathcal{K} \\
& \mu^s_{ij} \leq 0, \forall (ij) \in \mathcal{A}
\end{aligned} \tag{4.3}$$

The master problem now writes:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum_{s\in\mathcal{S}} p_s\theta^s \\
\text{s.t.} \quad & \theta^s \geq \sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s,u} - \lambda_{D_k,k}^{s,u}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s,u}, \forall s \in \mathcal{S}, \forall u \in \mathcal{U}^{(t-1)} \\
& 0 \geq \sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s,v} - \lambda_{D_k,k}^{s,v}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s,v}, \forall s \in \mathcal{S}, \forall v \in \mathcal{V}^{(t-1)} \\
& x_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{A}
\end{aligned}
\tag{4.4}
$$

where we define $\mathcal{U}^{(t-1)}$ and $\mathcal{V}^{(t-1)}$ with the following multi-cut Benders decomposition algorithm:

1. Initialize: $t = 1$, $\mathcal{U}(0) = \emptyset$, $\mathcal{V}(0) = \emptyset$
2. Solve master problem 4.4
   - If there is no solution, stop here: the problem is infeasible.
   - If the problem is unbounded, find $(\boldsymbol{x}^t, \theta^{1,t}, \ldots, \theta^{S,t})$ such that $\theta^{s,t} < -M$ where M is a large constant.
   - Otherwise, derive an optimal solution $(\boldsymbol{x}^t, \theta^{1,t}, \ldots, \theta^{S,t})$
3. For each scenario s,
   (a) solve the primal subproblem and check feasibility:
      - If it is unbounded, stop here: the problem is unbounded.
      - If it is finite, derive an optimal solution $\boldsymbol{y}^t$ and the corresponding dual solution $\boldsymbol{\lambda}^{s,u_t}, \boldsymbol{\mu}^{s,u_t}$
      - If infeasible, derive an extreme ray $\boldsymbol{\lambda}^{s,v_t}, \boldsymbol{\mu}^{s,v_t}$ such that $\sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s,v} - \lambda_{D_k,k}^{s,v}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s,v} > 0$ and add the corresponding feasibility cut $0 \geq \sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s,v_t} - \lambda_{D_k,k}^{s,v_t}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s,v_t}$ and update $\mathcal{V}^{(t-1)}$, $\mathcal{V}^{(t)} \leftarrow \mathcal{V}^{(t-1)} \cup \{v_t\}$.
   (b) Optimality condition:
      - If $\sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s} - \lambda_{D_k,k}^{s}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s} = \theta^{s,t}$, stop here, $(\boldsymbol{x}^t, \boldsymbol{y}^t)$ is optimal.
      - If not, add an optimality cut $\theta^s \geq \sum_{k\in\mathcal{K}} d_k^s \left(\lambda_{O_k,k}^{s,u_t} - \lambda_{D_k,k}^{s,u_t}\right) + \sum_{(i,j)\in\mathcal{A}} u_{ij}x_{ij}\mu_{ij}^{s,u_t}$, and update $\mathcal{U}^{(t)} \leftarrow \mathcal{U}^{(t-1)} \cup \{u_t\}$;
      - Return to Step 2.

b

Hardware is a Mac Book Pro 2019, 2.6 GHz 6-Core Intel Core i7, 16 GB 2667 MHz DDR4. Experimentally we find:

- Number of iterations: 21
- Number of optimality cuts: 10
- Number of feasibility cuts: 32
- Optimal solution: 2743.14

We obtain the following plot for the evolution of the upper bound and of the lower bound in Figure 4.1:

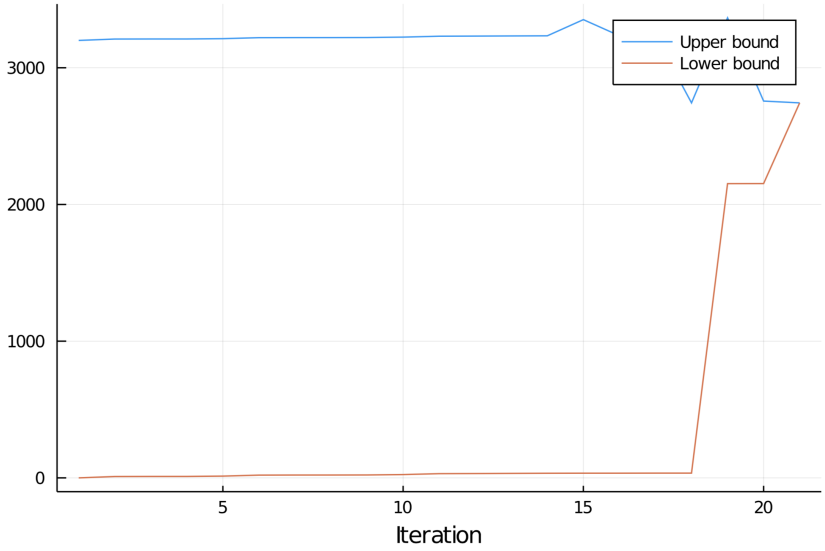


Figure 4.1: Evolution of upper and lower bounds across iterations.

We observe the following behaviors:

- For around 15 iterations, the two subproblems remain infeasible, and feasibility cuts are therefore added. In this phase, the upper bound can be equal to the maximum value we imposed (3200), and the lower bound increases very slowly.
- After these first iterations, for around 3 iterations several optimality cuts are added and the upper bound starts decreasing although it oscillates. The lower bound remains very small.
- For the last 3 iterations, the lower bound increases very quickly, until it reaches the lower bound at iteration 21.

Problem 2

a

We can reformulate the linear relaxation problem as follows:

$$\begin{aligned} \min \quad & \sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} c^A_{ap}x_{ap} + \theta \\ \text{s.t.} \quad & \sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}x_{ap} = 1, \forall i \in \mathcal{F} \\ & \sum_{p\in\mathcal{P}_a} x_{ap} = 1, \forall a \in \mathcal{A} \\ & x_{ap} \geq 0, \forall a \in \mathcal{A}, p \in \mathcal{P}_a \\ & \theta \geq Q(x) \end{aligned} \tag{4.5}$$

with

$$\begin{aligned} Q(x) = \quad \min \quad & \sum_{k\in\mathcal{K}}\sum_{q\in\mathcal{Q}_k} c^C_{kq}y_{kq} \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}}\sum_{q\in\mathcal{Q}_k} \delta^C_{iq}y_{kq} = 1, \forall i \in \mathcal{F} \\ & \sum_{k\in\mathcal{K}}\sum_{q\in\mathcal{Q}_k} \delta^C_{iq}\delta^C_{jq}y_{kq} \leq \sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap}, \forall (i,j) \in \mathcal{C} \\ & \sum_{q\in\mathcal{Q}_k} y_{kq} = 1, \forall k \in \mathcal{K} \\ & y_{kq} \geq 0, \forall k \in \mathcal{K}, q \in \mathcal{Q}_k \end{aligned} \tag{4.6}$$

We do not enforce the constraints $x \leq 1$ and $y \leq 1$ as they are directly implied by the other constraints.

The dual of the recourse function writes:

$$\begin{aligned} Q(x) = \quad \max \quad & \sum_{i\in\mathcal{F}} \alpha_i + \sum_{k\in\mathcal{K}} \beta_k + \sum_{(i,j)\in\mathcal{C}} \gamma_{ij}\left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap}\right) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{F}} \delta^C_{iq}\alpha_i + \beta_k + \sum_{(i,j)\in\mathcal{C}} \delta^C_{iq}\delta^C_{jq}\gamma_{ij} \leq c^C_{kq}, \forall k \in \mathcal{K}, q \in \mathcal{Q}_k \\ & \boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{F}|}, \boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{K}|}, \boldsymbol{\gamma} \leq 0 \end{aligned} \tag{4.7}$$

Putting everything together we now rewrite the master problem as:

$$
\begin{aligned}
\min & \sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} c^A_{ap}x_{ap} + \theta \\
\text{s.t.} & \sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}x_{ap} = 1, \forall i \in \mathcal{F} \\
& \sum_{p\in\mathcal{P}_a} x_{ap} = 1, \forall a \in \mathcal{A} \\
& \theta \geq \sum_{i\in\mathcal{F}} \alpha^u_i + \sum_{k\in\mathcal{K}} \beta^u_k + \sum_{(i,j)\in\mathcal{C}} \gamma^u_{ij} \left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap} \right), \forall u \in \mathcal{U}^{(t-1)} \\
& \sum_{i\in\mathcal{F}} \alpha^u_i + \sum_{k\in\mathcal{K}} \beta^u_k + \sum_{(i,j)\in\mathcal{C}} \gamma^u_{ij} \left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap} \right) \leq 0, \forall v \in \mathcal{V}^{(t-1)} \\
& x_{ap} \geq 0, \forall a \in \mathcal{A}, p \in \mathcal{P}_a
\end{aligned}
\tag{4.8}
$$

We obtain the following Benders decomposition algorithm (very similar to Problem 1):

1. Initialization: $t = 1$, $\mathcal{U}(0) = \emptyset$, $\mathcal{V}(0) = \emptyset$
2. We solve the master problem.
   - If there is no solution, we stop here: the linear relaxation is infeasible.
   - If the problem is unbounded, we find $(\boldsymbol{x}^t, \theta^t)$ such that $\theta^t < -M$ with M a large constant.
   - Otherwise, we derive an optimal solution $(\boldsymbol{x}^t, \theta^t)$.
3. Solve subproblem.
   - If it is unbounded, we stop here: the linear relaxation is unbounded.
   - If it is finite, we derive an optimal solution $\boldsymbol{y}^t$ and a dual solution $\boldsymbol{\alpha}^{u_t}, \boldsymbol{\beta}^{u_t}, \boldsymbol{\gamma}^{u_t}$
   - If it is infeasible, we derive an extreme ray $\boldsymbol{\alpha}^{v_t}, \boldsymbol{\beta}^{v_t}, \boldsymbol{\gamma}^{v_t}$ such that $\sum_{i\in\mathcal{F}} \alpha^u_i + \sum_{k\in\mathcal{K}} \beta^u_k + \sum_{(i,j)\in\mathcal{C}} \gamma^u_{ij} \left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap} \right) > 0$
4. Optimality condition:
   - If $\sum_{k\in\mathcal{K}}\sum_{q\in\mathcal{Q}_k} c^C_{kq}y_{kq} = \theta^t$, we stop here, $(\boldsymbol{x}^t, \boldsymbol{y}^t)$ is optimal;
   - Otherwise, add an optimality cut

     $$\theta \geq \sum_{i\in\mathcal{F}} \alpha^{u_t}_i + \sum_{k\in\mathcal{K}} \beta^{u_t}_k + \sum_{(i,j)\in\mathcal{C}} \gamma^{u_t}_{ij} \left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap} \right)$$

     and $\mathcal{U}^{(t)} \leftarrow \mathcal{U}^{(t-1)} \cup \{u_t\}$; or add a feasibility cut

     $$0 \geq \sum_{i\in\mathcal{F}} \alpha^{v_t}_i + \sum_{k\in\mathcal{K}} \beta^{v_t}_k + \sum_{(i,j)\in\mathcal{C}} \gamma^{v_t}_{ij} \left(\sum_{a\in\mathcal{A}}\sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp}x_{ap} \right)$$

and $\mathcal{V}^{(t)} \leftarrow \mathcal{V}^{(t-1)} \cup \{v_t\}$

- Go back to Step 2.

b

The problem at stake for the column generation algorithm is the following:

$$\begin{aligned} Q(x) = \quad \min \quad & \sum_{k\in\mathcal{K}} \sum_{q\in\mathcal{Q}_{k,0}} c^C_{kq} y_{kq} \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}} \sum_{q\in\mathcal{Q}_{k,0}} \delta^C_{iq}\delta^C_{jq} y_{kq} \leq \sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}_a} \delta^A_{ip}\delta^A_{jp} x_{ap}, \forall (i,j) \in \mathcal{C} \\ & \sum_{k\in\mathcal{K}} \sum_{q\in\mathcal{Q}_{k,0}} \delta^C_{iq} y_{kq} = 1, \forall i \in \mathcal{F} \\ & \sum_{q\in\mathcal{Q}_{k,0}} y_{kq} = 1, \forall k \in \mathcal{K} \\ & y_{kq} \geq 0, \forall k \in \mathcal{K}, q \in \mathcal{Q}_{k,0} \end{aligned} \tag{4.9}$$

The general principle is to do column generation on the route variables and follow the methodology seen in class for the generalized assignment problem. In our case, for a feasible solution, each crew is affected to at least one possible route. Therefore, we propose to consider at first only a subset of the available routes for each one of the crew $k \in \mathcal{K}$, and solve the problem with only this subset of routes. We then obtain an upper bound to the general problem.

Column generation algorithm The gist is to find which route to add in the master subproblem. Therefore, we decompose in the subproblem a route by the set of flights that composes it. Then, for each crew we compute the minimal reduced cost for all the other possible crew-route assignments. If reduced costs are all positive, the current solution is optimal. If not, we add in the master subproblem the variable corresponding to the lowest reduced cost.

The algorithm is as follows:

1. (Initialization) For each crew k, we generate the initial routes $\mathcal{Q}_{k,0}$.
2. We solve the problem 4.9 and should obtain the primal solution $\boldsymbol{y}^*$ and the dual solution $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*$.
3. We update $UB \leftarrow \sum_{k\in\mathcal{K}} \sum_{q\in\mathcal{Q}_{k,0}} c^C_{kq} y_{kq}$
4. We solve the general subproblem:

$$\bar{c}^* = \min_{k\in\mathcal{K}} \mathcal{P}_k - \beta^*_k \tag{4.10}$$

where $\mathcal{P}_k$ is the subproblem for each crew k:

$$\mathcal{P}_k = \min_{q\in\mathcal{Q}_k} c^C_{kq} - \sum_{i\in\mathcal{F}} \alpha^*_i \delta^C_{iq} - \sum_{(i,j)\in\mathcal{C}} \delta^C_{iq}\delta^C_{jq}\gamma^*_{ij} \tag{4.11}$$

Let k^* be the corresponding optimal column.

5. If $\bar{c}^* < 0$, do $\mathcal{Q}_{k^*,0} \leftarrow \mathcal{Q}_{k^*,0} \cup \{q^*\}$ where q^* is the route obtained by solving 4.11 and go back to 2 and repeat until termination.

6. Else, we stop and $\boldsymbol{y}^*$ is optimal.

c

Heuristic Branching Strategy We want to recover integer solutions and we propose a depth-only search heuristic as our goal is to obtain an integer solution as fast as possible, even if it not particularly good.

This heuristic should provide in the end an integer feasible solution to the restricted master problem.

Phase I We relax all integrality requirements, and the LP relaxation of the problem is solved to optimality by Benders decomposition and column generation as described in the two previous questions.

Phase II We retain all the cuts generated in the first phase, and we now reintroduce integrality constraints on the master problem aircraft path variables (x constraints) and solve the resulting mixed-integer problem by generating additional cuts. In this phase, we solve an IP master problem and LP primal subproblems.

The integer master problem must be solved again at each iteration of the Benders decomposition algorithm. We do not require to solve this problem to optimality at each iteration, because we propose to use the heuristic described previously to branch on the path variables and recover a feasible integer solution. We stop this heuristic once we have recovered an integer solution x for the first stage variables.

Phase III We now add integrality constraints on the subproblem for the crew path variables (y variables). We solve the integer subproblem once, while keeping the variable values of the master problem fixed, using values found in Phase II. We also branch directly on the path variables. Since originally the subproblem did not have the integrality property, the current integer subproblem may be infeasible for the given solution of the master problem, even if the original problem is feasible.

Comments The advantage of this approach is its fast runtime on the master problem. If we have a small integrality gap after solving using for example branch and bound, the method described can be particularly efficient. However, as mentioned the integer subproblem could be infeasible given the master problem. Therefore, we are not guaranteed to find a feasible solution to the overall problem, including the integer second-stage variables.

Problem 3

a

We can write:

$$\begin{aligned} z^* &= b^\top \pi_0 + \sum_{j=1}^{n} c'_j x_j^* \\ &= b^\top \pi_0 + \sum_{j=1}^{n} \left(c_j - \pi_0^\top a_j\right) x_j^* \\ &= b^\top \pi_0 + c^\top x^* - \pi_0^\top \sum_{j=1}^{n} a_j x_j^* \\ &= b^\top \pi_0 + c^\top x^* - \pi_0^\top b \\ &= c^\top x^* \end{aligned}$$

b

Let be π_0 a feasible dual solution (we consider we have it) and consider $\bar{z}$ an upper bound of (BIP). Since we have $\bar{z} \geq z^*$, we use the result of (3.a) to write:

$$\bar{z} - b^\top \pi_0 \geq \sum_{j=1}^{n} c'_j x_j^*$$

Since π_0 is a feasible dual solution, all reduced costs c'_j are positive since we included all variables in the problem. Therefore, if $c'_j > \bar{z} - b^\top \pi_0$, we cannot have $x_j = 1$ as it would break the previous inequality.

In conclusion, we propose the following variable fixing decision rule: if $c'_j > \bar{z} - b^\top \pi_0$, then we can fix $x_j^* = 0$.

c

The key idea is to fix some variables to 0 using the aforementioned rule to avoid generating those columns. However, the method of variable fixing requires the ability to identify a good UB for each iteration of solving the subproblem. If so, the acceleration strategy will be particularly effective. We propose the following strategy to accelerate the master problem solving:

Phase 1 We use an initial subset of columns $\mathcal{J}_0$ that easily give a feasible solution. Let x_0 be the primal solution we found and π_0 the corresponding dual solution. We therefore have the following upper bound $\bar{z} = c^T x_0$ and we can apply the aforementioned decision rule (3.b) to fix some of the variables in $\mathcal{J}_0$ to 0. We remove those fixed variables from the initial pool of columns.

Phase 2 Now, for a subsequent iteration t, with columns considered in $\mathcal{J}_t$, we consider we have access to a feasible primal solution x_t, and a dual solution π_t. We have the corresponding upper bound $c^T x_t$. Again with the decision rule, we fix some of the variables in $\mathcal{J}_t$ to 0 based on their reduced cost.

Comments This heuristic is a nice way to do column management as the set of considered columns can be reduced, allowing the master problem to be solved faster and more efficiently.

However, the dual solution that we use at each iteration may not be feasible for the entire problem. It means that in a later stage, we may need to add again a column that has been removed in order to get a final optimal solution.

We can also use the heuristic to accelerate the subproblems, instead of solving to optimality. We propose to go over the columns, and stop when we obtain one with negative reduced cost. We then include it in the set of considered columns.

As mentioned, the methodology will be particularly interesting if each dual solution obtained is not too far away from the final dual solution, to avoid cycling of removed/added columns.

In [22]:
```
using DataFrames, CSV
using JuMP, Gurobi
using LinearAlgebra, Random, Printf
using Plots

const GRB_ENV = Gurobi.Env()
```

```
Academic license - for non-commercial use only - expires 2021-07-08
```

Out[22]:
```
Gurobi.Env(Ptr{Nothing} @0x00007f867c9e1400, false, 0)
```

In [102…
```
arcs = CSV.read("Pb1_arcs.csv", DataFrame)
transport = CSV.read("Pb1_transport.csv", DataFrame)
demand = CSV.read("Pb1_demand.csv", DataFrame)
customers = CSV.read("Pb1_customer_OD.csv", DataFrame)
arcs_ = convert.(Int64,arcs[!,2:3])
```

Out[102… 90 rows × 2 columns

	x2	x3
	Int64	Int64
1	2	1
2	3	1
3	4	1
4	5	1
5	6	1
6	7	1
7	8	1
8	9	1
9	10	1
10	1	2
11	3	2
12	4	2
13	5	2
14	6	2
15	7	2
16	8	2
17	9	2
18	10	2
19	1	3
20	2	3
21	4	3
22	5	3
23	6	3
24	7	3
25	8	3
26	9	3
27	10	3
28	1	4

29	2	4
30	3	4
⋮	⋮	⋮

In [17]:
```
customers
```

Out[17]: 3 rows × 2 columns

	x1	x2
	Int64	Int64
1	1	9
2	4	2
3	8	2

1. Problem setup

In [107…
```
TIME_LIMIT = 90;
OPTIMALITY_GAP = 0.001;

"Solve problem using multi-cut Benders decomposition"
function solve_benders_multi(verbose::Bool=true)
    # define main problem
    MP = Model(() -> Gurobi.Optimizer(GRB_ENV));
    set_optimizer_attributes(MP, "TimeLimit" => 60, "MIPGap" => 1e-4, "Outp
    A = size(arcs, 1); K = size(customers, 1); S = size(demand, 2)
    @variable(MP, x[1:A], Bin)
    @variable(MP, θ[1:S] >= 0)
    @objective(MP, Min, sum(arcs[i,5] * x[i] for i=1:A) + sum(1/S * θ[s] fo
    lower_bound_all = []; upper_bound_all = []
    MP_time = []; SP_max_time = []; SP_time = []
    num_opt = 0; num_feas = 0
    while true
        # solve master problem
        push!(MP_time, @elapsed optimize!(MP))
        lower_bound_new = objective_value(MP)
        push!(lower_bound_all, lower_bound_new)
        x_MP = value.(MP[:x])
        # solve S subproblems
        obj_SP = zeros(S)
        SP_time_all = zeros(S)
        for s = 1:S

            SP_dual = Model(JuMP.optimizer_with_attributes(() -> Gurobi.Opt
                "OutputFlag" => 0, "DualReductions" => 0, "TimeLimit" => 60

            @variable(SP_dual, λ[1:10,1:K]);
            @variable(SP_dual, μ[1:A] <= 0);

            @objective(SP_dual, Max,
                       sum((λ[customers[k,1],k]-λ[customers[k,2],k]) * dema
                       sum(μ[i] * arcs[i,4] * x_MP[i] for i in 1:A))

            @constraint(SP_dual, [i in 1:A, k in 1:K],
                λ[arcs_[i,1],k] - λ[arcs_[i,2],k] + μ[i] <= transport[i,k])

            SP_time_all[s] = @elapsed optimize!(SP_dual)
            #obj_SP_dual = objective_value(SP_dual)
            λ_val = value.(SP_dual[:λ])
            μ_val = value.(SP_dual[:μ])
            if termination_status(SP_dual) == MOI.DUAL_INFEASIBLE # feasibl
                @constraint(MP, sum((λ_val[customers[k,1],k]-λ_val[customer
                       sum(μ_val[i] * arcs[i,4] * x[i] for i in 1:A) <= 0)
                obj_SP[s] = 3200
```

```
                num_feas += 1
            elseif termination_status(SP_dual) == MOI.OPTIMAL
                @constraint(MP, θ[s] >= sum((λ_val[customers[k,1],k]-λ_val
                       sum(μ_val[i] * arcs[i,4] * x[i] for i in 1:A))
                obj_SP[s] = objective_value(SP_dual)
                num_opt += 1
            end
        end
        push!(SP_max_time, maximum(SP_time_all))
        push!(SP_time, sum(SP_time_all))
        upper_bound_new = sum(arcs[i,5] * x_MP[i] for i=1:A) + sum(1/S * ob
        push!(upper_bound_all, upper_bound_new)
        verbose && @printf("UB: %.2f - LB: %.2f\n", upper_bound_all[end], l
        if sum(MP_time) + sum(SP_time) >= TIME_LIMIT ||
            (upper_bound_new-lower_bound_new)/lower_bound_new < OPTIMALITY_
            break
        end
    end
    return upper_bound_all, lower_bound_all, MP_time, SP_time, SP_max_time,
end
```

Out[107… solve_benders_multi

In [108…

```
upper_bound_all, lower_bound_all, MP_time, SP_time, SP_max_time, num_feas,
```

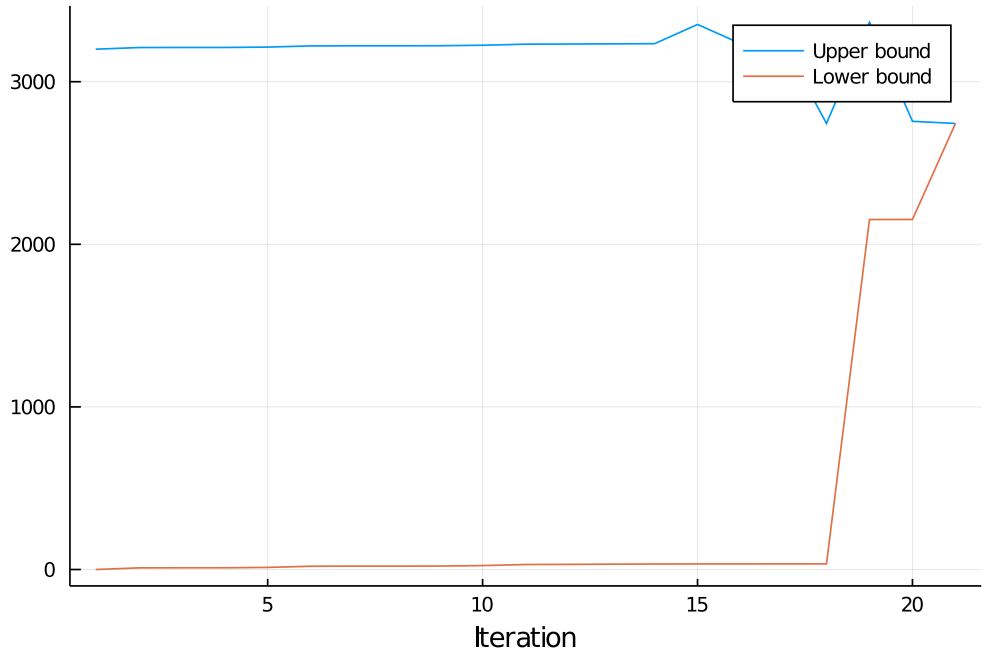
```
UB: 3200.00 - LB: 0.00
UB: 3210.05 - LB: 10.05
UB: 3210.55 - LB: 10.55
UB: 3210.56 - LB: 10.56
UB: 3212.90 - LB: 12.90
UB: 3220.11 - LB: 20.11
UB: 3220.74 - LB: 20.74
UB: 3220.81 - LB: 20.81
UB: 3221.25 - LB: 21.25
UB: 3224.19 - LB: 24.19
UB: 3230.80 - LB: 30.80
UB: 3231.69 - LB: 31.69
UB: 3232.80 - LB: 32.80
UB: 3233.77 - LB: 33.77
UB: 3352.03 - LB: 34.25
UB: 3234.29 - LB: 34.29
UB: 3234.74 - LB: 34.74
UB: 2743.14 - LB: 34.75
UB: 3365.28 - LB: 2152.35
UB: 2756.38 - LB: 2152.85
UB: 2743.14 - LB: 2743.14
```

Out[108… (Any[3200.0, 3210.054529448178, 3210.5504810699804, 3210.562429355936, 3212
.898361166845, 3220.1136273655734, 3220.742821014491, 3220.807068585424, 32
21.2507209222485, 3224.190843426695 … 3231.6882584851733, 3232.8006372684
135, 3233.773629273072, 3352.0328794045035, 3234.286055186306, 3234.7413244
966756, 2743.136793777782, 3365.27728915555, 2756.381203528829, 2743.136793
777782], Any[0.0, 10.054529448177977, 10.550481069980382, 10.5624293559357,
12.89836116684521, 20.113627365573514, 20.742821014490755, 20.8070685854242
7, 21.250720922248476, 24.19084342669534 … 31.688258485173435, 32.8006372
6841344, 33.77362927307213, 34.24994134409088, 34.28605518630586, 34.741324
49667572, 34.75327278263104, 2152.350838312871, 2152.854169751412, 2743.136
7937777823], Any[0.001021085, 0.002473057, 0.000699174, 0.001314428, 0.0014
64564, 0.005083717, 0.00157934, 0.002331034, 0.001769716, 0.002026514 … 0
.002012163, 0.002877235, 0.006162246, 0.003313874, 0.002469903, 0.002317671
, 0.002410215, 0.002512856, 0.002510676, 0.003399851], Any[0.005480275, 0.0
04052873, 0.004486044999999999, 0.004129187, 0.006008623, 0.004787151, 0.00
5361995, 0.0046422600000000005, 0.004833864, 0.005578787 … 0.003817553, 0
.005045758, 0.0038845250000000002, 0.0035313410000000003, 0.004731394, 0.00
4545399, 0.0036984210000000003, 0.003261647, 0.003215611, 0.004845708], Any
[0.002845987, 0.002088845, 0.002348809, 0.002199612, 0.003261186, 0.0029602
61, 0.002686592, 0.002463351, 0.002689996, 0.003098361 … 0.001990776, 0.0
02670213, 0.00246172, 0.001961768, 0.002371545, 0.002298368, 0.002084203, 0
.00166272, 0.001688792, 0.003139663], 32, 10)

In [109…

```
plot([upper_bound_all lower_bound_all], label=["Upper bound" "Lower bound"
```

Out[109…



In [106…

```
@show length(upper_bound_all)
@show num_opt
@show num_feas
```

```
length(upper_bound_all) = 21
num_opt = 10
num_feas = 32
```

Out[106… 32