Exercise Sheet 3

Integer Programming
Academic Year 2022/23: Quarter 4

Lecturer: Christopher Hojny

TU/e

Facility Location

Consider a company that can produce goods in up to n factories $F = \{1, \dots, n\}$; if a factory $j \in F$ is opened, this causes a fixed cost f_j. Moreover, the company has m customers $C = \{1, \dots, m\}$ that demand the goods produced in the factories. To satisfy their demand, the company delivers goods directly from the factories to the customers; if the entire demand of customer $i \in C$ is satisfied by goods from factory $j \in F$, then a cost of c_{ij} applies.

We assume that a factory can produce arbitrarily many goods. If we introduce variables

- $x_j \in \{0, 1\}$, $j \in F$, indicating whether factory j is open;
- $y_{ij} \in \mathbb{R}_+$ encoding the portion of the demand of customer $i \in C$ that is satisfied by factory j;

our problem can be modeled as

$$\begin{aligned} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} y_{ij} + \sum_{j=1}^{n} f_j x_j & \\ \sum_{j=1}^{n} y_{ij} \geq 1, & \qquad i \in C, \\ y_{ij} - x_j \leq 0, & \qquad i \in C,\ j \in F. \end{aligned}$$

Your task is to help the company finding a set of factories to open and distributing the customers' demands to the open factories that minimizes total cost. To this end, we provided a template file on CANVAS as well as a few test instances. The template file already contains a routine for reading an instance of the the factory location problem in the section "Auxiliary Functions". Extend this template file by your own routines and optimization models. Please put your own methods in the corresponding sections (Auxiliary Functions, Exercise 1, Exercise 2,...) of this file and submit your solution via CANVAS.

Exercise 1 (The Uncapacitated Model) 1.5 points

Note that once the set of open factories has been selected, the above mixed-integer linear problem turns into a linear program. Implement a Benders' decomposition approach for solving the uncapacitated factory location problem, where the master problem contains only the original x-variables and subproblem objective encoding variables.

You are not required to implement the separation of Benders cuts via a callback; a while-loop that solves the problem and possibly adds a cut is sufficient.

Hints:

- To ensure that there always exists a feasible assignment of customers to the open factories suggested by the master problem, you may consider to add $\sum_{j=1}^{n} x_j \geq 1$ to the master problem.
- To be able to access information on unbounded instances, you have to use the parameter setting `model.Params.InfUnbdInfo = True`. If Gurobi detects that a problem is infeasible, you can get information on an unbounded ray by calling `var.UnbdRay` for each variable `var` in your problem.

- To terminate the Benders loop, you might need to ask Gurobi for its current status, see https://www.gurobi.com/documentation/9.0/refman/optimization_status_codes.html.

Exercise 2 (Avoiding to Solve Linear Programs) 1+2 points

In a naive implementation of the above Benders' decomposition approach, we need to solve a linear program in each iteration. The aim of this exercise is to show that this is not necessary for our model, because we get a solution to this linear program essentially for free.

Consider the uncapacitated problem (P) as defined above.

1. If x is fixed to an arbitrary binary vector, P reduces to a linear program. Find the dual problem of this linear program, using α-variables for the first family of inequalities and β-variables for the second family of inequalities.
2. Let $\bar{x} \in \{0,1\}^F$, where

$$O(\bar{x}) = \{j \in F : \bar{x}_j = 1\} \quad \text{and} \quad C(\bar{x}) = \{j \in F : \bar{x}_j = 0\},$$

i.e., $O(\bar{x})$ encodes the set of opened and $C(\bar{y})$ the set of closed factories. Show that

$$\alpha_i = \min_{j \in O(\bar{x})} c_{ij},$$

$$\beta_{ij} = \begin{cases} 0, & j \in O(\bar{x}), \\ \max\{\alpha_i - c_{ij}, 0\}, & j \in C(\bar{x}), \end{cases}$$

is an optimal solution of the dual problem.

Hint: The duality theorem of linear programming might be helpful.

Polyhedral Classifiers

The idea of Benders' decomposition can also be used to find polyhedral classifiers. Let $X, Y \subseteq \mathbb{Z}^n$ be a pair of positive and negative data points and assume we want to find a polyhedral classifier with at most K inequalities. Then, a compact model to find a polyhedral classifier is given by

$$\min \sum_{i=1}^{K} u_i \tag{1a}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad i \in \{1, \dots, K\},\ x \in X, \tag{1b}$$

$$\sum_{j=1}^{n} a_{ij} y_j \geq b_i + \varepsilon - M(1 - s_{yi}), \qquad i \in \{1, \dots, K\},\ y \in Y, \tag{1c}$$

$$\sum_{i=1}^{K} s_{yi} \geq 1, \qquad y \in Y, \tag{1d}$$

$$u_i \in \{0, 1\}, \qquad i \in \{1, \dots, K\}, \tag{1e}$$

$$s_{yi} \in \{0, 1\}, \qquad y \in Y,\ i \in \{1, \dots, K\}, \tag{1f}$$

$$-1 \leq a_{ij} \leq 1, \qquad i \in \{1, \dots, K\},\ j \in \{1, \dots, n\}, \tag{1g}$$

$$-B \leq b_i \leq B, \qquad i \in \{1, \dots, n\}, \tag{1h}$$

where M is a sufficiently large constant and $B = n \max\{|x_i| : x \in X,\ i \in \{1, \dots, n\}\}$. For instance, $M = B + \varepsilon + n \max\{|y_i| : y \in Y,\ i \in \{1, \dots, n\}\}$ is a valid choice. Note that u models which of the potential K inequalities are selected, s models which negative data point is separated by which inequality, and (a, b) models the different inequalities.

Exercise 3 (Combinatorial Benders cuts) 1.5+1+2+1 points

Observe that once the u- and s-variables are fixed, the problem reduces to a linear program. That is, we can apply Benders' decomposition to solve it.

1. Implement a Benders' decomposition approach for solving the polyhedral classifier problem.
   In your implementation, take into account that the problem decomposes into the K different classes for potential inequalities. That is, instead of solving one single Benders subproblem, solve K smaller Benders subproblems and transfer the cuts from one class to the other classes.
2. Note that the continuous variables do not contribute to the objective. That is, if the first stage variables u and s are fixed, we only need to decide whether the remaining subproblem is feasible. If it is feasible, we have found an optimal solution. Otherwise, we need to forbid the assignment of s-variables. Since the s-variables are binary, this can be achieved by so-called "no good" cuts:
   Let $\bar{s} \in \{0, 1\}^{Y \times \{1, \dots, K\}}$. Prove that the only binary point that violates
   $$\sum_{(y,i):\bar{s}_{yi}=0} s_{yi} + \sum_{(y,i):\bar{s}_{yi}=1} (1 - s_{yi}) \geq 1$$
   is $\bar{s}$. That is, the no good cut can be used to forbid the current assignment of s-variables.

3. The no good cut is a very weak inequality as it forbids exactly one assignment of s-variables. One way of strengthening the inequality is to remove terms from the left-hand side while still providing a valid inequality.

   Develop a strategy to strengthen the no good cuts. Describe your strategy in detail and discuss possible advantages and disadvantages of your approach for finding a strengthened cut.

4. Implement a Benders decomposition purely based on (strengthened) no good cuts. Compare the running time of this procedure with the running time of your implementation for classical Benders cuts.

Submission and Grading

To submit your solution on CANVAS,

- extend the provided template file by adding your own methods and data structures to the corresponding sections of the template;
- for the theoretical part, prepare a PDF file containing your solution;
- in your PDF for the theoretical part, describe how you derive and separate the Benders cuts for both exercises—use the same variable names in the PDF document and your code to make checking correctness of the code easier;
- submit your solution until June 18, 20:00.
- work in groups of up to 3 students, list all group members in the header section of the template file.

To achieve a top score, your solution has to satisfy the following criteria:

- Answers to all exercises are provided;
- The implementation and answers to theoretical questions are correct;
- The master problems are implemented efficiently, i.e., you do not build them from scratch each time a new inequality has been separated. Instead, you have to add the inequality to the already existing master problems.
- The subproblems are implemented efficiently, i.e., you do not build them from scratch each time a new inequality is separated. Instead, you have to adapt the objective function.