Hands-on Large Scale Optimization in Python

From Beginning to Giving Up

Kunlei Lian

2023-02-18

Table of contents

Preface

1 Environment Setup

In this chapter, we explain the steps needed to set up Python and Google OR-Tools. All the steps below are based on MacBook Air with M1 chip and macOS Ventura 13.1.

1.1 Install Homebrew

The first tool we need is Homebrew, 'the Missing Package Manager for macOS (or Linux)', and it can be accessed at https://brew.sh/. To install Homebrew, just copy the command below and run it in the Terminal.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.s
```

We can then use the `brew --version` command to check the installed version. On my system, it shows the info below.

```
  ~/ brew --version
Homebrew 3.6.20
Homebrew/homebrew-core (git revision 5f1582e4d55; last commit 2023-02-05)
Homebrew/homebrew-cask (git revision fa3b8a669d; last commit 2023-02-05)
```

1.2 Install Anaconda

Since there are several Python versions available for our use and we may end up having multiple Python versions installed on our machine, it is important to use a consistent environment to work on our project in. Anaconda is a package and environment manager for Python and it provides easy-to-use tools to facilitate our data science needs. To install Anaconda, run the below command in the Terminal.

```
  ~/ brew install anaconda
```

After the installation is done, we can use `conda --version` to verify whether it is available on our machine or not.

```
  ~/ conda --version
conda 23.1.0
```

1.3 Create a Conda Environment

Now we will create a Conda environment named ‘ortools’. Execute the below command in the Terminal, which effectively creates the required environment with Python version 3.10.

```
  ~/ conda create -n ortools python=3.10
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/homebrew/anaconda3/envs/test

  added / updated specs:
    - python=3.10


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    setuptools-67.4.0          |     pyhd8ed1ab_0         567 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         567 KB

The following NEW packages will be INSTALLED:

  bzip2              conda-forge/osx-arm64::bzip2-1.0.8-h3422bc3_4
  ca-certificates    conda-forge/osx-arm64::ca-certificates-2022.12.7-h4653dfc_0
  libffi             conda-forge/osx-arm64::libffi-3.4.2-h3422bc3_5
  libsqlite          conda-forge/osx-arm64::libsqlite-3.40.0-h76d750c_0
  libzlib            conda-forge/osx-arm64::libzlib-1.2.13-h03a7124_4
  ncurses            conda-forge/osx-arm64::ncurses-6.3-h07bb92c_1
  openssl            conda-forge/osx-arm64::openssl-3.0.8-h03a7124_0
  pip                conda-forge/noarch::pip-23.0.1-pyhd8ed1ab_0
  python             conda-forge/osx-arm64::python-3.10.9-h3ba56d0_0_cpython
  readline           conda-forge/osx-arm64::readline-8.1.2-h46ed386_0
  setuptools          conda-forge/noarch::setuptools-67.4.0-pyhd8ed1ab_0
  tk                  conda-forge/osx-arm64::tk-8.6.12-he1e0b03_0
  tzdata              conda-forge/noarch::tzdata-2022g-h191b570_0
  wheel               conda-forge/noarch::wheel-0.38.4-pyhd8ed1ab_0
  xz                  conda-forge/osx-arm64::xz-5.2.6-h57fd34a_0


Proceed ([y]/n)?
```

Type 'y' to proceed and Conda will create the environment for us. We can use `cnoda env list` to show all the created environments on our machine:

```
  ~/ conda env list
# conda environments:
#
base                     /opt/homebrew/anaconda3
ortools                   /opt/homebrew/anaconda3/envs/ortools
```

Note that we need to manually activate an environemnt in order to use it: `conda activate ortools`. On my machine, the activated environment `ortools` will appear in the beginning of my prompt.

```
  ~/ conda activate ortools
(ortools)   ~/
```

1.4 Install Google OR-Tools

As of this writing, the latest version of Google OR-Tools is 9.5.2237, and we can install it in our newly created environment using the command `pip install ortools==9.5.2237`. We can use `conda list` to verify whether it is available in our environment.

```
(ortools)   ~/ conda list
# packages in environment at /opt/homebrew/anaconda3/envs/ortools:
#
# Name                    Version                   Build  Channel
absl-py                   1.4.0                    pypi_0    pypi
bzip2                     1.0.8                h3422bc3_4    conda-forge
ca-certificates           2022.12.7            h4653dfc_0    conda-forge
libffi                    3.4.2                h3422bc3_5    conda-forge
libsqlite                 3.40.0               h76d750c_0    conda-forge
```

```
libzlib                   1.2.13               h03a7124_4    conda-forge
ncurses                   6.3                  h07bb92c_1    conda-forge
numpy                     1.24.2                   pypi_0    pypi
openssl                   3.0.8                h03a7124_0    conda-forge
ortools                   9.5.2237                 pypi_0    pypi
pip                       23.0.1             pyhd8ed1ab_0    conda-forge
protobuf                  4.22.0                   pypi_0    pypi
python                    3.10.9      h3ba56d0_0_cpython    conda-forge
readline                  8.1.2                h46ed386_0    conda-forge
setuptools                67.4.0             pyhd8ed1ab_0    conda-forge
tk                        8.6.12               he1e0b03_0    conda-forge
tzdata                    2022g                h191b570_0    conda-forge
wheel                     0.38.4             pyhd8ed1ab_0    conda-forge
xz                        5.2.6                h57fd34a_0    conda-forge
```

Now we have Python and Google OR-Tools ready, we can start our next journey.

2 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

3 Environment Setup

In this chapter, we explain the steps needed to set up Python and Google OR-Tools. All the steps below are based on MacBook Air with M1 chip and macOS Ventura 13.1.

3.1 Install Homebrew

The first tool we need is Homebrew, 'the Missing Package Manager for macOS (or Linux)', and it can be accessed at https://brew.sh/. To install Homebrew, just copy the command below and run it in the Terminal.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.s
```

We can then use the `brew --version` command to check the installed version. On my system, it shows the info below.

```
  ~/ brew --version
Homebrew 3.6.20
Homebrew/homebrew-core (git revision 5f1582e4d55; last commit 2023-02-05)
Homebrew/homebrew-cask (git revision fa3b8a669d; last commit 2023-02-05)
```

3.2 Install Anaconda

Since there are several Python versions available for our use and we may end up having multiple Python versions installed on our machine, it is important to use a consistent environment to work on our project in. Anaconda is a package and environment manager for Python and it provides easy-to-use tools to facilitate our data science needs. To install Anaconda, run the below command in the Terminal.

```
  ~/ brew install anaconda
```

After the installation is done, we can use `conda --version` to verify whether it is available on our machine or not.

```
 ~/ conda --version
conda 23.1.0
```

3.3 Create a Conda Environment

Now we will create a Conda environment named ‘ortools’. Execute the below command in the Terminal, which effectively creates the required environment with Python version 3.10.

```
 ~/ conda create -n ortools python=3.10
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/homebrew/anaconda3/envs/test

  added / updated specs:
    - python=3.10


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    setuptools-67.4.0          |     pyhd8ed1ab_0         567 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         567 KB

The following NEW packages will be INSTALLED:

  bzip2              conda-forge/osx-arm64::bzip2-1.0.8-h3422bc3_4
  ca-certificates    conda-forge/osx-arm64::ca-certificates-2022.12.7-h4653dfc_0
  libffi             conda-forge/osx-arm64::libffi-3.4.2-h3422bc3_5
  libsqlite          conda-forge/osx-arm64::libsqlite-3.40.0-h76d750c_0
  libzlib            conda-forge/osx-arm64::libzlib-1.2.13-h03a7124_4
  ncurses            conda-forge/osx-arm64::ncurses-6.3-h07bb92c_1
  openssl            conda-forge/osx-arm64::openssl-3.0.8-h03a7124_0
  pip                conda-forge/noarch::pip-23.0.1-pyhd8ed1ab_0
  python             conda-forge/osx-arm64::python-3.10.9-h3ba56d0_0_cpython
  readline           conda-forge/osx-arm64::readline-8.1.2-h46ed386_0
  setuptools         conda-forge/noarch::setuptools-67.4.0-pyhd8ed1ab_0
  tk                 conda-forge/osx-arm64::tk-8.6.12-he1e0b03_0
  tzdata             conda-forge/noarch::tzdata-2022g-h191b570_0
  wheel              conda-forge/noarch::wheel-0.38.4-pyhd8ed1ab_0
  xz                 conda-forge/osx-arm64::xz-5.2.6-h57fd34a_0


Proceed ([y]/n)?
```

Type 'y' to proceed and Conda will create the environment for us. We can use `cnoda env list` to show all the created environments on our machine:

```
  ~/ conda env list
# conda environments:
#
base                     /opt/homebrew/anaconda3
ortools                   /opt/homebrew/anaconda3/envs/ortools
```

Note that we need to manually activate an environemnt in order to use it: `conda activate ortools`. On my machine, the activated environment `ortools` will appear in the beginning of my prompt.

```
  ~/ conda activate ortools
(ortools)  ~/
```

3.4 Install Google OR-Tools

As of this writing, the latest version of Google OR-Tools is 9.5.2237, and we can install it in our newly created environment using the command `pip install ortools==9.5.2237`. We can use `conda list` to verify whether it is available in our environment.

```
(ortools)  ~/ conda list
# packages in environment at /opt/homebrew/anaconda3/envs/ortools:
#
# Name                    Version                   Build  Channel
absl-py                   1.4.0                    pypi_0    pypi
bzip2                     1.0.8                h3422bc3_4    conda-forge
ca-certificates           2022.12.7            h4653dfc_0    conda-forge
libffi                    3.4.2                h3422bc3_5    conda-forge
libsqlite                 3.40.0               h76d750c_0    conda-forge
```

```
libzlib                   1.2.13               h03a7124_4    conda-forge
ncurses                   6.3                  h07bb92c_1    conda-forge
numpy                     1.24.2                   pypi_0    pypi
openssl                   3.0.8                h03a7124_0    conda-forge
ortools                   9.5.2237                 pypi_0    pypi
pip                       23.0.1             pyhd8ed1ab_0    conda-forge
protobuf                  4.22.0                   pypi_0    pypi
python                    3.10.9          h3ba56d0_0_cpython    conda-forge
readline                  8.1.2                h46ed386_0    conda-forge
setuptools                67.4.0             pyhd8ed1ab_0    conda-forge
tk                        8.6.12               he1e0b03_0    conda-forge
tzdata                    2022g                h191b570_0    conda-forge
wheel                     0.38.4             pyhd8ed1ab_0    conda-forge
xz                        5.2.6                h57fd34a_0    conda-forge
```

Now we have Python and Google OR-Tools ready, we can start our next journey.

Part I

Benders Decomposition

4 Benders Decomposition

In this chapter, we will explain the theories behind Benders decomposition and demonstrate its usage on a trial linear programming problem. Keep in mind that Benders decomposition is not limited to solving linear programming problems. In fact, it is one of the most powerful techniques to solve some large-scale mixed-integer linear programming problems.

In the following sections, we will go through the critical steps during the decomposition process when applying the algorithm on optimization problems represented in standard forms. This is important as it helps build up the intuition of when we should consider applying Benders decomposition to a problem at hand. Often times, recognizing the applicability of Benders decomposition is the most important and challenging step when solving an optimization problem. Once we know that the problem structure is suitable to solve via Benders decomposition, it is straightforward to follow the decomposition steps and put it into work.

Generally speaking, Benders decomposition is a good solution candidate when the resulting problem is much easier to solve if some of the variables in the original problem are fixed. We will illustrate this point using an example in the following sections. In the optimization world, the first candidate that should come to mind when we say a problem is easy to solve is a linear programming formulation, which is indeed the case in Benders decomposition applications.

4.1 The Decomposition Logic

To explain the reasoning of Benders decomposition, let us look at the standard form of linear programming problems that involve two vector variables, $\mathbf{x}$ and $\mathbf{y}$. Let p and q indicate the dimensions of $\mathbf{x}$ and $\mathbf{y}$, respectively. Below is the original problem $\mathbf{P}$ we intend to solve.

$$
(\mathbf{P}) \qquad
\begin{aligned}
\text{min.} \quad & \mathbf{c}^T\mathbf{x} + \mathbf{f}^T\mathbf{y} && (4.1)\\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b} && (4.2)\\
& \mathbf{x} \geq 0, \mathbf{y} \geq 0 && (4.3)
\end{aligned}
$$

In this formulation, $\mathbf{c}$ and $\mathbf{f}$ in the objective function represent the cost coefficients associated with decision variables $\mathbf{x}$ and $\mathbf{y}$, respectively. Both of them are column vectors of corresponding dimensions. In the constraints, matrix $\mathbf{A}$ is of dimension $m \times p$, and matrix $\mathbf{B}$ is of dimension $m \times q$. $\mathbf{b}$ is a column vector of dimension m.

Suppose the variable **y** is a complicating variable in the sense that the resulting problem is substantially easier to solve if the value of **y** is fixed. In this case, we could rewrite problem **P** as the following form:

$$\begin{aligned} \text{min.} \quad & \mathbf{f}^T\mathbf{y} + g(\mathbf{y}) && (4.4) \\ \text{s.t.} \quad & \mathbf{y} \geq 0 && (4.5) \end{aligned}$$

where $g(\mathbf{y})$ is a function of **y** and is defined as the subproblem **SP** of the form below:

$$(\mathbf{SP}) \qquad \begin{aligned} \text{min.} \quad & \mathbf{c}^T\mathbf{x} && (4.6) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} - \mathbf{B}\mathbf{y} && (4.7) \\ & \mathbf{x} \geq 0 && (4.8) \end{aligned}$$

Note that the **y** in constraint (4.7) takes on some known values when the problem is solved and the only decision variable in the above formulation is **x**. The dual problem of **SP**, **DSP**, is given below.

$$(\mathbf{DSP}) \qquad \begin{aligned} \text{max.} \quad & (\mathbf{b} - \mathbf{B}\mathbf{y})^T\mathbf{u} && (4.9) \\ \text{s.t.} \quad & \mathbf{A}^T\mathbf{u} \leq \mathbf{c} && (4.10) \\ & \mathbf{u} \text{ unrestricted} && (4.11) \end{aligned}$$

A key characteristic of the above **DSP** is that its solution space does not depend on the value of **y**, which only affects the objective function. According to the Minkowski's representation theorem, any $\bar{\mathbf{u}}$ satisfying the constraints (4.10) can be expressed as

$$\bar{\mathbf{u}} = \sum_{j \in \mathbf{J}} \lambda_j \mathbf{u}_j^{point} + \sum_{k \in \mathbf{K}} \mu_k \mathbf{u}_k^{ray} \qquad (4.12)$$

where $\mathbf{u}_j^{point}$ and $\mathbf{u}_k^{ray}$ represent an extreme point and extreme ray, respectively. In addition, $\lambda_j \geq 0$ for all $j \in \mathbf{J}$ and $\sum_{j \in \mathbf{J}} \lambda_j = 1$, and $\mu_k \geq 0$ for all $k \in \mathbf{K}$. It follows that the **DSP** is equivalent to

$$\begin{aligned}
\text{max.} \quad & (\mathbf{b} - \mathbf{B}\mathbf{y})^T(\sum_{j \in \mathbf{J}} \lambda_j \mathbf{u}_j^{point} + \sum_{k \in \mathbf{K}} \mu_k \mathbf{u}_k^{ray}) && (4.13)\\
\text{s.t.} \quad & \sum_{j \in \mathbf{J}} \lambda_j = 1 && (4.14)\\
& \lambda_j \geq 0, \ \forall j \in \mathbf{J} && (4.15)\\
& \mu_k \geq 0, \ \forall k \in \mathbf{K} && (4.16)
\end{aligned}$$

We can therefore conclude that

- The **DSP** becomes unbounded if any $\mathbf{u}_k^{ray}$ exists such that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} > 0$. Note that an unbounded **DSP** implies an infeasible **SP** and to prevent this from happening, we have to ensure that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0$ for all $k \in \mathbf{K}$.
- If an optimal solution to **DSP** exists, it must occur at one of the extreme points. Let g denote the optimal objective value, it follows that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g$ for all $j \in \mathbf{J}$.

Based on this idea, the **DSP** can be reformulated as follows:

$$\begin{aligned}
\text{min.} \quad & g && (4.17)\\
\text{s.t.} \quad & (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0, \ \forall j \in \mathbf{J} && (4.18)\\
& (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g, \ \forall k \in \mathbf{K} && (4.19)\\
& j \in \mathbf{J}, k \in \mathbf{K} && (4.20)
\end{aligned}$$

Constraints (4.18) are called **Benders feasibility cuts**, while constraints (4.19) are called **Benders optimality cuts**. Now we are ready to define the Benders Master Problem (**BMP**) as follows:

$$(\mathbf{BMP}) \qquad \begin{aligned}
\text{min.} \quad & \mathbf{f}^T \mathbf{y} + g && (4.21)\\
\text{s.t.} \quad & (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0, \ \forall j \in \mathbf{J} && (4.22)\\
& (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g, \ \forall k \in \mathbf{K} && (4.23)\\
& j \in \mathbf{J}, k \in \mathbf{K}, \mathbf{y} \geq 0 && (4.24)
\end{aligned}$$

Typically J and K are too large to enumerate upfront and we have to work with subsets of them, denoted by J_s and K_s, respectively. Hence we have the following Restricted Benders Master Problem (**RBMP**):

$$
\begin{aligned}
(\mathbf{RBMP}) \qquad\qquad \text{min.} \quad & \mathbf{f}^T\mathbf{y} + g && (4.25)\\
\text{s.t.} \quad & (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0, \;\; \forall j \in \mathbf{J}_s && (4.26)\\
& (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g, \;\; \forall k \in \mathbf{K}_s && (4.27)\\
& j \in \mathbf{J}, k \in \mathbf{K}, \mathbf{y} \geq 0 && (4.28)
\end{aligned}
$$

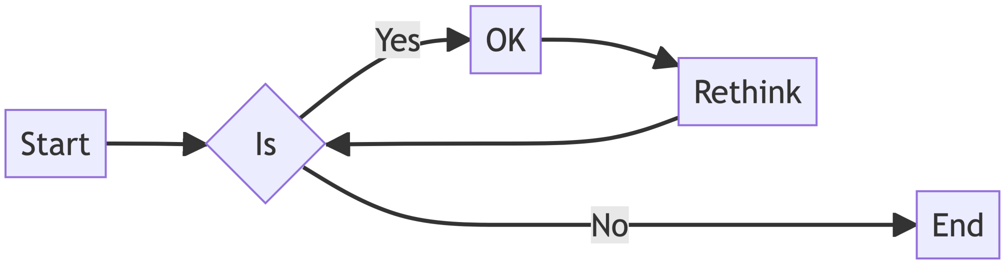

Figure 4.1: Benders decomposition workflow

4.2 Solving Linear Programming Problems with Benders Decomposition

In this section, we use Benders decomposition to solve several linear programming (LP) problems in order to demonstrate the decomposition logic, especially how the restricted Benders master problem interacts with the subproblem in an iterative approach to reach final optimality. Most linear programs could be solved efficiently nowadays by either open source or commercial solvers without resorting to any decomposition approaches. However, by working through the example problems in the following sections, we aim to showcase the implementation details when applying Benders decomposition algorithm on real problems, which helps solidify our understanding of Benders decomposition. Hopefully, by the end of this chapter, we will build up enough intuition as well as hands-on experience such that we are ready to tackle most involved problems in the following chapters.

In the following sections, we employ several steps to illustrate the problem solving process of Benders decomposition.

- We will first create two linear programming solvers based on Gurobi and SCIP that can solve any linear programs defined in the standard form. They are used in later section to validate the correctness of the solutions produced by Benders decomposition.
- Next, we use a specific linear program and give the corresponding RBMP and DSP to prepare for the implementations.

- Then, we will solve the example linear program step by step by examining the outputs of the RBMP and DSP to decided the next set of actions.
- Futhermore, a holistic Benders decomposition implementation is then developed to solve the example linear program.
- Following the previous step, a more generic Benders decomposition implementation is created.
- Then, we will examine an alternative implementation using Gurobi callback functions.
- We will also provide an implementation based on SCIP.
- In the final section, we will do several benchmarking testing.

4.2.1 LP solvers based on Gurobi and SCIP

We aim to use Benders decomposition to solve several linear programming problems in the following sections. To do that, we intentionally decompose the LP problem under consideration into two sets, one set of *complicating* variables and the other set containing the remaining variables. In order to validate the correctness of the results obtained by Benders decomposition, we implement two additional ways of solving the target linear programming problems directly. The first option is based on the Gurobi API in python and the other is based on the open source solve SCIP. The two implementations defined here assume the LP problems under consideration follow the below format.

$$\begin{aligned} \text{min.} \quad & \mathbf{c}^T\mathbf{x} && (4.29)\\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} && (4.30)\\ & \mathbf{x} \geq 0 && (4.31) \end{aligned}$$

Listing 4.1 defines a solver for LP problems using Gurobi. It takes three constructor parameters:

- `obj_coeff`: this corresponds to the objective coefficients $\mathbf{c}$.
- `constr_mat`: this refers to the constraint matrix $\mathbf{A}$.
- `rhs`: this is the right-hand side $\mathbf{b}$.

Inside the constructor `__init__()`, a solver environment `_env` is first created and then used to initialize a model object `_model`. The input parameters are then used to create decision variables `_vars`, constraints `_constrs` and objective function respectively. The `optimize()` function simply solves the problem and shows the solving status. Finally, the `clean_up()` function frees up the computing resources.

Listing 4.2 presents an LP solver implementation in class `LpSolverSCIP` using SCIP. The constructor requires the same of parameters as defined in `LpSolverGurobi`. The model building process is similar with minor changes when required to create decision variables, constraints and the objective function.

Listing 4.3 generates a LP problem with 20 decision variables and 5 constraints.

Listing 4.4 solves the generated LP using `LpSolverGurobi` and the solver output in Listing 4.5 shows that an optimal solution was found with objective value of 46.61.

Listing 4.6 solves the same LP problem using SCIP and not surprisingly, the same optimal objective value was found, as shown in Listing 4.7. This is not exciting, as it only indicates that the two solvers agree on the optimal solution on such a small LP problem as expected. However, they will become more useful in the following sections when we use them to validate our Benders decomposition results.

4.2.2 A serious LP problem that cannot wait to be decomposed!

The linear program we examine here is devoid of any practical meaning and is solely used to demonstrate the solution process of Benders decomposition. The problem is stated below, in which $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2)$ are the decision variables. We assume that $\mathbf{y}$ is the complicating variable.

$$\begin{aligned} \text{min.} \quad & 8x_1 + 12x_2 + 10x_3 + 15y_1 + 18y_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 + 2x_3 + 4y_1 + 5y_2 = 300 \\ & 4x_1 + 2x_2 + 3x_3 + 2y_1 + 3y_2 = 220 \\ & x_i \geq 0, \ \forall i = 1, \cdots, 3 \\ & y_i \geq 0, \ \forall j = 1, 2 \end{aligned}$$

In this example, $\mathbf{c}^T = (8, 12, 10)$, $\mathbf{f}^T = (15, 18)$ and $\mathbf{b}^T = (300, 220)$. In addition,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 2 \\ 4 & 2 & 3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 4 & 5 \\ 2 & 3 \end{bmatrix}$$

We first use Gurobi to identify its optimal solution.

```python
import gurobipy as gp
from gurobipy import GRB

# Create a new model
env = gp.Env(empty=True)
env.setParam('OutputFlag', 0)
env.start()
```

```python
model = gp.Model(env=env, name="original_problem")

# Decision variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name='x1')
x2 = model.addVar(vtype=GRB.CONTINUOUS, name='x2')
x3 = model.addVar(vtype=GRB.CONTINUOUS, name='x3')
y1 = model.addVar(vtype=GRB.CONTINUOUS, name='y1')
y2 = model.addVar(vtype=GRB.CONTINUOUS, name='y2')

# Objective function
model.setObjective(8*x1 + 12*x2 + 10*x3 + 15*y1 + 18*y2,
                   GRB.MINIMIZE)

# Constraints
model.addConstr(2*x1 + 3*x2 + 2*x3 + 4*y1 + 5*y2 == 300)
model.addConstr(4*x1 + 2*x2 + 3*x3 + 2*y1 + 3*y2 == 220)

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print("Optimal solution found!")
    print(f'x1 = {x1.X:.2f}')
    print(f'x2 = {x2.X:.2f}')
    print(f'x3 = {x3.X:.2f}')
    print(f'y1 = {y1.X:.2f}')
    print(f'y2 = {y2.X:.2f}')
    print(f"Total cost: {model.objVal:.2f}")
else:
    print("No solution found.")

# Close the Gurobi environment
model.dispose()
env.dispose()
```

The optimal solution and objective value are as follows.

~~~
```{python}
Optimal solution found!
x1 = 14.29
x2 = 0.00
~~~

```
x3 = 0.00
y1 = 0.00
y2 = 54.29
Total cost: 1091.43
```
```

4.2.3 Benders decomposition on paper

We first state the subproblem as follows:

$$
(\mathbf{SP}) \qquad \begin{aligned}
\text{min.} \quad & 8x_1 + 12x_2 + 10x_3 \\
\text{s.t.} \quad & 2x_1 + 3x_2 + 2x_3 = 300 - 4y_1 - 5y_2 \\
& 4x_1 + 2x_2 + 3x_3 = 220 - 2y_1 - 3y_2 \\
& x_i \geq 0, \ \forall i = 1, \cdots, 3
\end{aligned}
$$

We define two dual variables u_1 and u_2 to associate with the two constraints in the subproblem. The dual subproblem could then be stated as follows:

$$
(\mathbf{DSP}) \qquad \begin{aligned}
\text{max.} \quad & (300 - 4y_1 - 5y_2)u_1 + (220 - 2y_1 - 3y_2)u_2 \\
\text{s.t.} \quad & 2u_1 + 4u_2 \leq 8 \\
& 3u_1 + 2u_2 \leq 12 \\
& 2u_1 + 3u_2 \leq 10 \\
& u_1, u_2 \text{ unrestricted}
\end{aligned}
$$

The *RBMP* can be stated as:

$$
(\mathbf{RBMP}) \qquad \begin{aligned}
\text{min.} \quad & 15y_1 + 18y_2 + g \\
\text{s.t.} \quad & y_1, y_2 \geq 0 \\
& g \leq 0
\end{aligned}
$$

4.2.4 Benders decomposition step by step

In this section, we will solve the linear program step by step using Gurobi. To this end, we first import the necessary libraries and create an environment `env`.
```

```
#  output: false
import numpy as np
import gurobipy as gp
from gurobipy import GRB

env = gp.Env('benders')
env.setParam('OutputFlag', 0)
```

Next, we initialize several algorithm parameters, specifically, we use `lb` and `ub` to represent the lower and upper bounds of the solution. The `eps` is defined as a small number to decide whether the searching process should stop.

The remaining codes aim to create the restricted master Benders problem indicated by `rbmp`. Note that it only has the y and g variables and the objective function, there is no constraint added to the model yet.

```
# parameters
lb = -GRB.INFINITY
ub = GRB.INFINITY
eps = 1.0e-5

# create restricted Benders master problem
rbmp = gp.Model(env=env, name='RBMP')

# create decision variables
y1 = rbmp.addVar(vtype=GRB.CONTINUOUS, lb=0, name='y1')
y2 = rbmp.addVar(vtype=GRB.CONTINUOUS, lb=0, name='y2')
g = rbmp.addVar(vtype=GRB.CONTINUOUS, lb=0, name='g')

# create objective
rbmp.setObjective(15*y1 + 18*y2 + g, GRB.MINIMIZE)
```

We then define the model in Gurobi to solve the dual subproblem, represented by `dsp`. It consists of two decision variables `u1` and `u2`. The constraints are created in lines 12 - 14.

```
# create dual subproblem
dsp = gp.Model(env=env, name='DSP')

# create decision variables
u1 = dsp.addVar(vtype=GRB.CONTINUOUS, name='u1')
u2 = dsp.addVar(vtype=GRB.CONTINUOUS, name='u2')

```

```
# create objective function
dsp.setObjective(300*u1 + 220*u2)

# create constraints
dsp.addConstr(2*u1 + 4*u2 <= 8, name='c1')
dsp.addConstr(3*u1 + 2*u2 <= 12, name='c2')
dsp.addConstr(2*u1 + 3*u2 <= 10, name='c3')

dsp.update()
```

In the very first iteration, we solve the **RBMP**, as shown in the following code snippet.

```
rbmp.optimize()

if rbmp.status == GRB.OPTIMAL:
    print(f'optimal solution found!')

    y1_opt = y1.X
    y2_opt = y2.X
    g_opt = g.X
    lb = np.max([lb, rbmp.objVal])

    print(f'optimal obj: {rbmp.objVal:.2f}')
    print(f'y1 = {y1_opt:.2f}')
    print(f'y2 = {y2_opt:.2f}')
    print(f'g = {g_opt:.2f}')
    print(f'lb={lb}, ub={ub}')
elif rbmp.status == GRB.INFEASIBLE:
    print(f'original problem is infeasible!')
```

Now we have obtained an optimal solution $(\bar{y_1}, \bar{y_2}, \bar{g}) = (0, 0, 0)$, which also provides a new lower bound to our problem. We now feed the values of $\bar{y_1}$ and $\bar{y_2}$ into the Benders subproblem **(SP)**:

```
dsp.setObjective((300-4*y1_opt-5*y2_opt)*u1 +
                 (220-2*y1_opt-3*y2_opt)*u2,
                 GRB.MAXIMIZE)
dsp.update()
dsp.optimize()

if dsp.status == GRB.OPTIMAL:
    u1_opt = u1.X
    u2_opt = u2.X

    print(f'optimal obj = {dsp.objVal:.2f}')
    print(f'u1 = {u1_opt:.2f}')
    print(f'u2 = {u2_opt:.2f}')
    ub = np.min([ub, 15*y1_opt + 18*y2_opt + dsp.objVal])
    print(f'lb={lb}, ub={ub}')
elif dsp.Status == GRB.UNBOUNDED:
    # add feasibility cut
    pass
else:
    pass
```

We see that the dual subproblem has an optimal solution. Note that in line 15, the upper bound of the problem is updated.

Since the optimal objective value of the subproblem turns out to be 1200 and is greater than $\bar{g} = 0$, which implies that an optimality cut is needed to make sure that the variable g in the restricted Benders master problem reflects this newly obtained information from the subproblem.

```
rbmp.addConstr((300-4*y2-5*y2)*u1_opt
               + (220-2*y1-3*y2)*u2_opt <= g,
               name='c3')
rbmp.update()
rbmp.optimize()

if rbmp.status == GRB.OPTIMAL:
    print(f'optimal solution found!')

    y1_opt = y1.X
    y2_opt = y2.X
    g_opt = g.X
    lb = np.max([lb, rbmp.objVal])

    print(f'optimal obj: {rbmp.objVal:.2f}')
    print(f'y1 = {y1_opt:.2f}')
    print(f'y2 = {y2_opt:.2f}')
    print(f'g = {g_opt:.2f}')
    print(f'lb={lb}, ub={ub}')
elif rbmp.status == GRB.INFEASIBLE:
    print(f'original problem is infeasible!')
```

Now we solve the subproblem again with the newly obtained solution $(\bar{y_1}, \bar{y_2}, \bar{g}) = (0, 33.33, 0)$.

```
dsp.setObjective((300 - 4*y1_opt - 5*y2_opt) * u1
                 + (220 - 2*y1_opt - 3*y2_opt) * u2,
                 GRB.MAXIMIZE)
dsp.update()
dsp.optimize()

if dsp.status == GRB.OPTIMAL:
    u1_opt = u1.X
    u2_opt = u2.X

    print(f'optimal obj = {dsp.objVal:.2f}')
    print(f'u1 = {u1_opt:.2f}')
    print(f'u2 = {u2_opt:.2f}')
    ub = np.min([ub, 15*y1_opt + 18*y2_opt + dsp.objVal])
    print(f'lb={lb}, ub={ub}')
elif dsp.status == GRB.UNBOUNDED:
    print(f'dual subproblem is unbounded!')
```

Since the optimal objective value of the subproblem, 533.33, is still bigger than $\bar{g} = 0$, an optimality cut is needed. In the below code snippet, we add the new cut and solve the restricted Benders master problem again.

```
rbmp.addConstr((300 - 4*y1 - 5*y2) * u1_opt + (220 - 2*y1 - 3*y2) * u2_opt <= g, name='c3')
rbmp.update()
rbmp.optimize()

if rbmp.status == GRB.OPTIMAL:
    print(f'optimal solution found!')

    y1_opt = y1.X
    y2_opt = y2.X
    g_opt = g.X
    lb = np.max([lb, rbmp.objVal])

    print(f'optimal obj: {rbmp.objVal:.2f}')
    print(f'y1 = {y1_opt:.2f}')
    print(f'y2 = {y2_opt:.2f}')
    print(f'g = {g_opt:.2f}')
    print(f'lb={lb}, ub={ub}')
elif rbmp.status == GRB.INFEASIBLE:
    print(f'original problem is infeasible!')
```

Note that a new lower bound is obtained after solving the master problem. Since there is still a large gap between the lower bound and upper bound, we continue solving the subproblem.

```
dsp.setObjective((300 - 4*y1_opt - 5*y2_opt) * u1 + (220 - 2*y1_opt - 3*y2_opt) * u2, GRB.MA
dsp.update()
dsp.optimize()

if dsp.status == GRB.OPTIMAL:
    u1_opt = u1.X
    u2_opt = u2.X

    print(f'optimal obj = {dsp.objVal:.2f}')
    print(f'u1 = {u1_opt:.2f}')
    print(f'u2 = {u2_opt:.2f}')
    ub = np.min([ub, 15*y1_opt + 18*y2_opt + dsp.objVal])
    print(f'lb={lb}, ub={ub}')
elif dsp.status == GRB.UNBOUNDED:
    print(f'dual subproblem is unbounded!')
```

Now the upper bound is reduced to 1133.33, but the subproblem optimal solution is still bigger than the value of $\bar{g} = 0$.

```
rbmp.addConstr((300 - 4*y1 - 5*y2) * u1_opt + (220 - 2*y1 - 3*y2) * u2_opt <= g, name='c3')
rbmp.update()
rbmp.optimize()

if rbmp.status == GRB.OPTIMAL:
    print(f'optimal solution found!')

    y1_opt = y1.X
    y2_opt = y2.X
    g_opt = g.X
    lb = np.max([lb, rbmp.objVal])

    print(f'optimal obj: {rbmp.objVal:.2f}')
    print(f'y1 = {y1_opt:.2f}')
    print(f'y2 = {y2_opt:.2f}')
    print(f'g = {g_opt:.2f}')
    print(f'lb={lb}, ub={ub}')
elif rbmp.status == GRB.INFEASIBLE:
    print(f'original problem is infeasible!')
```

```
dsp.setObjective((300 - 4*y1_opt - 5*y2_opt) * u1 + (220 - 2*y1_opt - 3*y2_opt) * u2, GRB.MA
dsp.update()
dsp.optimize()

if dsp.status == GRB.OPTIMAL:
    u1_opt = u1.X
    u2_opt = u2.X

    print(f'optimal obj = {dsp.objVal:.2f}')
    print(f'u1 = {u1_opt:.2f}')
    print(f'u2 = {u2_opt:.2f}')
    ub = np.min([ub, 15*y1_opt + 18*y2_opt + dsp.objVal])
    print(f'lb={lb}, ub={ub}')
elif dsp.status == GRB.UNBOUNDED:
    print(f'dual subproblem is unbounded!')
```

Now the gap between the lower bound and upper bound is reduced to 0, the problem completes.

4.2.5 Putting it together

Certainly we don't want to manually control the interaction between the master problem and subproblem to find the optimal solution. Therefore, in this section, we will put every together to come up with a control flow to help us identify the optimal solution automatically.

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np
from enum import Enum

class OptStatus(Enum):
    OPTIMAL = 0
    UNBOUNDED = 1
    INFEASIBLE = 2
    ERROR = 3
```

```python
class MasterSolver:

    def __init__(self, env):
        self._model = gp.Model(env=env, name='RBMP')

        # create decision variables
        self._y1 = self._model.addVar(vtype=GRB.CONTINUOUS, lb=0, name='y1')
        self._y2 = self._model.addVar(vtype=GRB.CONTINUOUS, lb=0, name='y2')
        self._g = self._model.addVar(vtype=GRB.CONTINUOUS, lb=0, name='g')

        # create objective
        self._model.setObjective(15*self._y1 + 18*self._y2 + self._g, GRB.MINIMIZE)

        self._opt_obj = None
        self._opt_y1 = None
        self._opt_y2 = None
        self._opt_g = None

    def solve(self) -> OptStatus:
        print('-' * 50)
        print(f'Start solving master problem.')
        self._model.optimize()

        opt_status = None
        if self._model.status == GRB.OPTIMAL:
            opt_status = OptStatus.OPTIMAL
            self._opt_obj = self._model.objVal
            self._opt_y1 = self._y1.X
            self._opt_y2 = self._y2.X
            self._opt_g = self._g.X
            print(f'\tmaster problem is optimal.')
            print(f'\topt_obj={self._opt_obj:.2f}')
            print(f'\topt_y1={self._opt_y1:.2f}, opt_y2={self._opt_y2:.2f}, opt_g={self._opt_
        elif self._model.status == GRB.INFEASIBLE:
            print(f'\tmaster problem is infeasible.')
            opt_status = OptStatus.INFEASIBLE
        else:
            print(f'\tmaster problem encountered error.')
            opt_status = OptStatus.ERROR

        print(f'Finish solving master problem.')
        print('-' * 50)
        return opt_status

    def add_feasibility_cut(self, opt_u1, opt_u2) -> None:
        self._model.addConstr((300 - 4*self._y1 - 5*self._y2) * opt_u1 +
                              (220 - 2*self._y1 - 3*self._y2) * opt_u2 <= 0)
        print(f'Benders feasibility cut added!')

    def add_optimality_cut(self, opt_u1, opt_u2) -> None:
        self._model.addConstr((300 - 4*self._y1 - 5*self._y2) * opt_u1 +
                              (220 - 2*self._y1 - 3*self._y2) * opt_u2 <= self._g)
        print(f'Benders optimality cut added!')

    def clean_up(self):
        self._model.dispose()

    @property
    def opt_obj(self):
        return self._opt_obj

    @property
    def opt_y1(self):
        return self._opt_y1

    @property
    def opt_y2(self):
        return self._opt_y2

    @property
    def opt_g(self):
        return self._g
```

```
class DualSubprobSolver:

    def __init__(self, env):
        self._model = gp.Model(env=env, name='DSP')

        # create decision variables
        self._u1 = self._model.addVar(vtype=GRB.CONTINUOUS, name='u1')
        self._u2 = self._model.addVar(vtype=GRB.CONTINUOUS, name='u2')

        # create constraints
        self._model.addConstr(2*self._u1 + 4*self._u2 <= 8, name='c1')
        self._model.addConstr(3*self._u1 + 2*self._u2 <= 12, name='c2')
        self._model.addConstr(2*self._u1 + 3*self._u2 <= 10, name='c3')

        self._model.setObjective(1, GRB.MAXIMIZE)
        self._model.update()

        self._opt_obj = None
        self._opt_u1 = None
        self._opt_u2 = None

    def solve(self):
        print('-' * 50)
        print(f'Start solving dual subproblem.')
        self._model.optimize()

        status = None
        if self._model.status == GRB.OPTIMAL:
            self._opt_obj = self._model.objVal
            self._opt_u1 = self._u1.X
            self._opt_u2 = self._u2.X
            status = OptStatus.OPTIMAL
            print(f'\tdual subproblem is optimal.')
            print(f'\topt_obj={self._opt_obj:.2f}')
            print(f'\topt_y1={self._opt_u1:.2f}, opt_y2={self._opt_u2:.2f}')
        elif self._model.status == GRB.UNBOUNDED:
            status = OptStatus.UNBOUNDED
        else:
            status = OptStatus.ERROR

        print(f'Finish solving dual subproblem.')
        print('-' * 50)
        return status

    def update_objective(self, opt_y1, opt_y2):
        self._model.setObjective((300-4*opt_y1-5*opt_y2)*self._u1 + (220-2*opt_y1-3*opt_y2)*s
        print(f'dual subproblem objective updated!')

    def clean_up(self):
        self._model.dispose()

    @property
    def opt_obj(self):
        return self._opt_obj

    @property
    def opt_u1(self):
        return self._opt_u1

    @property
    def opt_u2(self):
        return self._opt_u2
```

```
class BendersDecomposition:

    def __init__(self, master_solver, dual_subprob_solver):
        self._master_solver = master_solver
        self._dual_subprob_solver = dual_subprob_solver


    def optimize(self) -> OptStatus:
        eps = 1.0e-5
        lb = -np.inf
        ub = np.inf

        while True:
            # solve master problem
            master_status = self._master_solver.solve()
            if master_status == OptStatus.INFEASIBLE:
                return OptStatus.INFEASIBLE

            # update lower bound
            lb = np.max([lb, self._master_solver.opt_obj])
            print(f'Bounds: lb={lb:.2f}, ub={ub:.2f}')

            opt_y1 = self._master_solver.opt_y1
            opt_y2 = self._master_solver.opt_y2

            # solve subproblem
            self._dual_subprob_solver.update_objective(opt_y1, opt_y2)
            dsp_status = self._dual_subprob_solver.solve()

            if dsp_status == OptStatus.OPTIMAL:
                # update upper bound
                opt_obj = self._dual_subprob_solver.opt_obj
                ub = np.min([ub, 15*opt_y1 + 18*opt_y2 + opt_obj])
                print(f'Bounds: lb={lb:.2f}, ub={ub:.2f}')

                if ub - lb <= eps:
                    break

                opt_u1 = self._dual_subprob_solver.opt_u1
                opt_u2 = self._dual_subprob_solver.opt_u2
                self._master_solver.add_optimality_cut(opt_u1, opt_u2)
            elif dsp_status == OptStatus.UNBOUNDED:
                opt_u1 = self._dual_subprob_solver.opt_u1
                opt_u2 = self._dual_subprob_solver.opt_u2
                self._master_solver.add_feasibility_cut(opt_u1, opt_u2)
```

```
env = gp.Env('benders')
env.setParam("OutputFlag",0)
master_solver = MasterSolver(env)
dual_subprob_solver = DualSubprobSolver(env)

benders_decomposition = BendersDecomposition(master_solver, dual_subprob_solver)
benders_decomposition.optimize()
```

```
Set parameter Username
Set parameter LogFile to value "benders"
--------------------------------------------------
Start solving master problem.
    master problem is optimal.
    opt_obj=0.00
    opt_y1=0.00, opt_y2=0.00, opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=0.00, ub=inf
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=1200.00
    opt_y1=4.00, opt_y2=0.00
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=0.00, ub=1200.00
```

```
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
    master problem is optimal.
    opt_obj=1080.00
    opt_y1=0.00, opt_y2=60.00, opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=1080.00, ub=1200.00
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=80.00
    opt_y1=0.00, opt_y2=2.00
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=1080.00, ub=1160.00
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
    master problem is optimal.
    opt_obj=1091.43
    opt_y1=0.00, opt_y2=54.29, opt_g=114.29
Finish solving master problem.
--------------------------------------------------
Bounds: lb=1091.43, ub=1160.00
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=114.29
    opt_y1=0.00, opt_y2=2.00
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=1091.43, ub=1091.43
```

4.2.6 A generic solver

In this section, we will create a more generic Benders decomposition based solver for linear programming problems.

```
class GenericLpMasterSolver:

    def __init__(self, f: np.array, B: np.array, b: np.array):
        # save data
        self._f = f
        self._B = B
        self._b = b

        # env and model
        self._env = gp.Env('MasterEnv')
        # self._env.setParam("OutputFlag",0)
        self._model = gp.Model(env=self._env, name='MasterSolver')

        # create variables
        self._num_y_vars = len(f)
        self._y = self._model.addVars(self._num_y_vars, lb=0, vtype=GRB.CONTINUOUS, name='y'
        self._g = self._model.addVar(vtype=GRB.CONTINUOUS, lb=0, name='g')

        # create objective
        self._model.setObjective(gp.quicksum(self._f[i] * self._y.get(i)
                                             for i in range(self._num_y_vars)) + self._g,
                                 GRB.MINIMIZE)
        self._model.update()

        self._opt_obj = None
        self._opt_obj_y = None
        self._opt_y = None
        self._opt_g = None

    def solve(self) -> OptStatus:
        print('-' * 50)
        print(f'Start solving master problem.')
        print(self._model.display())
        self._model.optimize()

        opt_status = None
        if self._model.status == GRB.OPTIMAL:
            opt_status = OptStatus.OPTIMAL
            self._opt_obj = self._model.objVal
            self._opt_y = {
                i: self._y.get(i).X
                for i in range(self._num_y_vars)
            }
            self._opt_g = self._g.X
            self._opt_obj_y = self._opt_obj - self._opt_g
            print(f'\tmaster problem is optimal.')
            print(f'\topt_obj={self._opt_obj:.2f}')
            print(f'\topt_g={self._opt_g:.2f}')
            # for i in range(self._num_y_vars):
            #     print(f'\topt_y{i}={self._opt_y.get(i)}')
        elif self._model.status == GRB.INFEASIBLE:
            print(f'\tmaster problem is infeasible.')
            opt_status = OptStatus.INFEASIBLE
        else:
            print(f'\tmaster problem encountered error.')
            opt_status = OptStatus.ERROR

        print(f'Finish solving master problem.')
        print('-' * 50)
        return opt_status

    def add_feasibility_cut(self, opt_u: dict) -> None:
        constr_expr = [
            opt_u.get(u_idx) * (self._b[u_idx] - gp.quicksum(self._B[u_idx][j] * self._y.get
                                                   for j in range(self._num_y_vars)))
            for u_idx in opt_u.keys()
        ]
        self._model.addConstr(gp.quicksum(constr_expr) <= 0)
        print(f'Benders feasibility cut added!')

    def add_optimality_cut(self, opt_u: dict) -> None:
        constr_expr = [
            opt_u.get(u_idx) * (self._b[u_idx] - gp.quicksum(self._B[u_idx][j] * self._y.get
                                                   for j in range(self._num_y_vars)))
            for u_idx in opt_u.keys()
        ]
        self._model.addConstr(gp.quicksum(constr_expr) <= self._g)
        self._model.update()
        print(self._model.display())
        print(f'Benders optimality cut added!')

    def clean_up(self):
        self._model.dispose()
        self._env.dispose()
    @property
    def f(self):
        return self._f

    @property
    def opt_obj(self):
        return self._opt_obj

    @property
    def opt_obj_y(self):
        return self._opt_obj_y

    @property
    def opt_y(self):
        return self._opt_y

    @property
    def opt_g(self):
        return self._g
```

```
class GenericLpSubprobSolver:

    def __init__(self, A: np.array, c: np.array, B: np.array, b: np.array):
        # save data
        self._A = A
        self._c = c
        self._b = b
        self._B = B

        # env and model
        self._env = gp.Env('SubprobEnv')
        self._env.setParam("OutputFlag",0)
        self._model = gp.Model(env=self._env, name='SubprobSolver')

        # create variables
        self._num_vars = len(b)
        self._u = self._model.addVars(self._num_vars, vtype=GRB.CONTINUOUS, name='u')

        # create constraints
        for c_idx in range(len(c)):
            self._model.addConstr(gp.quicksum(A[:,c_idx][i] * self._u.get(i)
                                            for i in range(len(b))) <= c[c_idx])

        self._opt_obj = None
        self._opt_u = None
        self._extreme_ray = None

    def solve(self):
        print('-' * 50)
        print(f'Start solving dual subproblem.')
        self._model.setParam(GRB.Param.DualReductions, 0)
        self._model.setParam(GRB.Param.InfUnbdInfo, 1)
        self._model.optimize()

        status = None
        if self._model.status == GRB.OPTIMAL:
            self._opt_obj = self._model.objVal
            self._opt_u = {
                i: self._u.get(i).X
                for i in range(self._num_vars)
            }
            status = OptStatus.OPTIMAL
            print(f'\tdual subproblem is optimal.')
            print(f'\topt_obj={self._opt_obj:.2f}')
            # for i in range(self._num_vars):
            #     print(f'\topt_u{i}={self._opt_u.get(i)}')
        elif self._model.status == GRB.UNBOUNDED:
            status = OptStatus.UNBOUNDED
            self._extreme_ray = {
                i: self._u.get(i).UnbdRay
                for i in range(self._num_vars)
            }
            print(f'dual subproblem is unbounded')
            # for i in range(self._num_vars):
            #     print(f'\topt_u{i}={self._extreme_ray.get(i)}')
        else:
            status = OptStatus.ERROR

        print(f'Finish solving dual subproblem.')
        print('-' * 50)
        return status

    def update_objective(self, opt_y: dict):
        obj_expr = [
            self._u.get(u_idx) * (self._b[u_idx] - sum(self._B[u_idx][j] * opt_y.get(j)
                                                        for j in range(len(opt_y))))
            for u_idx in range(self._num_vars)
        ]
        self._model.setObjective(gp.quicksum(obj_expr), GRB.MAXIMIZE)
        print(f'dual subproblem objective updated!')

    def clean_up(self):
        self._model.dispose()
        self._env.dispose()

    @property
    def opt_obj(self):
        return self._opt_obj

    @property
    def opt_u(self):
        return self._opt_u

    @property
    def extreme_ray(self):
        return self._extreme_ray
```

```
class GenericBendersSolver:

    def __init__(self, master_solver, dual_subprob_solver):
        self._master_solver = master_solver
        self._dual_subprob_solver = dual_subprob_solver


    def optimize(self,) -> OptStatus:
        eps = 1.0e-5
        lb = -np.inf
        ub = np.inf

        while True:
            # solve master problem
            master_status = self._master_solver.solve()
            if master_status == OptStatus.INFEASIBLE:
                return OptStatus.INFEASIBLE
```

```python
            # update lower bound
            lb = np.max([lb, self._master_solver.opt_obj])
            print(f'Bounds: lb={lb:.2f}, ub={ub:.2f}')

            opt_y = self._master_solver.opt_y

            # solve subproblem
            self._dual_subprob_solver.update_objective(opt_y)
            dsp_status = self._dual_subprob_solver.solve()

            if dsp_status == OptStatus.OPTIMAL:
                # update upper bound
                opt_obj = self._dual_subprob_solver.opt_obj
                opt_obj_y = self._master_solver.opt_obj_y
                ub = np.min([ub, opt_obj_y + opt_obj])
                print(f'Bounds: lb={lb:.2f}, ub={ub:.2f}')

                if ub - lb <= eps:
                    break

                opt_u = self._dual_subprob_solver.opt_u
                self._master_solver.add_optimality_cut(opt_u)
            elif dsp_status == OptStatus.UNBOUNDED:
                extreme_ray = self._dual_subprob_solver.extreme_ray
                self._master_solver.add_feasibility_cut(extreme_ray)
```

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np

c = np.array([8, 12, 10])
f = np.array([15, 18])
A = np.array([
    [2, 3, 2],
    [4, 2, 3]
])
B = np.array([
    [4, 5],
    [2, 3],
])
b = np.array([300, 220])
```

```
master_solver = GenericLpMasterSolver(f, B, b)
dual_subprob_solver = GenericLpSubprobSolver(A, c, B, b)

benders_solver = GenericBendersSolver(master_solver, dual_subprob_solver)
benders_solver.optimize()
```

```
Set parameter Username
Set parameter LogFile to value "MasterEnv"
Set parameter Username
Set parameter LogFile to value "SubprobEnv"
--------------------------------------------------
Start solving master problem.
Minimize
  15.0 y[0] + 18.0 y[1] + g
Subject To
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 0 rows, 3 columns and 0 nonzeros
Model fingerprint: 0xb00b2c2a
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [1e+00, 2e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 0 rows and 3 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  0.000000000e+00
    master problem is optimal.
    opt_obj=0.00
    opt_g=0.00
    opt_y0=0.0
    opt_y1=0.0
Finish solving master problem.
```

```
--------------------------------------------------
Bounds: lb=0.00, ub=inf
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=1200.00
    opt_u0=4.0
    opt_u1=0.0
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=0.00, ub=1200.00
Minimize
  15.0 y[0] + 18.0 y[1] + g
Subject To
  R0: -16.0 y[0] + -20.0 y[1] + -1.0 g <= -1200
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
  15.0 y[0] + 18.0 y[1] + g
Subject To
  R0: -16.0 y[0] + -20.0 y[1] + -1.0 g <= -1200
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 1 rows, 3 columns and 3 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 2e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+03, 1e+03]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.500000e+02   0.000000e+00      0s
       1    1.0800000e+03   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  1.080000000e+03
    master problem is optimal.
    opt_obj=1080.00
    opt_g=0.00
    opt_y0=0.0
    opt_y1=60.0
Finish solving master problem.
--------------------------------------------------
Bounds: lb=1080.00, ub=1200.00
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=80.00
    opt_u0=0.0
    opt_u1=2.0
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=1080.00, ub=1160.00
Minimize
  15.0 y[0] + 18.0 y[1] + g
Subject To
  R0: -16.0 y[0] + -20.0 y[1] + -1.0 g <= -1200
  R1: -4.0 y[0] + -6.0 y[1] + -1.0 g <= -440
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
  15.0 y[0] + 18.0 y[1] + g
Subject To
  R0: -16.0 y[0] + -20.0 y[1] + -1.0 g <= -1200
  R1: -4.0 y[0] + -6.0 y[1] + -1.0 g <= -440
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 2 rows, 3 columns and 6 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 2e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+02, 1e+03]
```

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0800000e+03   4.000000e+01   0.000000e+00      0s
       1    1.0914286e+03   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  1.091428571e+03
    master problem is optimal.
    opt_obj=1091.43
    opt_g=114.29
    opt_y0=0.0
    opt_y1=54.285714285714285
Finish solving master problem.
--------------------------------------------------
Bounds: lb=1091.43, ub=1160.00
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=114.29
    opt_u0=0.0
    opt_u1=2.0
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=1091.43, ub=1091.43
```

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np

c = np.array([1, 1, 1, 1, 1, 1])
f = np.array([1, 1, 1, 1])
A = np.array([
    [1, 1, 1, 1, 1, 1]
])
B = np.array([
    [1, 1, 1, 1]
])
b = np.array([1])

master_solver = GenericLpMasterSolver(f, B, b)
dual_subprob_solver = GenericLpSubprobSolver(A, c, B, b)
```

```python
benders_solver = GenericBendersSolver(master_solver, dual_subprob_solver)
benders_solver.optimize()
```

```
Set parameter Username
Set parameter LogFile to value "MasterEnv"
Set parameter Username
Set parameter LogFile to value "SubprobEnv"
--------------------------------------------------
Start solving master problem.
Minimize
  y[0] + y[1] + y[2] + y[3] + g
Subject To
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 0 rows, 5 columns and 0 nonzeros
Model fingerprint: 0xbadf3d1d
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 0 rows and 5 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  0.000000000e+00
    master problem is optimal.
    opt_obj=0.00
    opt_g=0.00
    opt_y0=0.0
    opt_y1=0.0
    opt_y2=0.0
    opt_y3=0.0
Finish solving master problem.
--------------------------------------------------
```

```
Bounds: lb=0.00, ub=inf
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=1.00
    opt_u0=1.0
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=0.00, ub=1.00
Minimize
  y[0] + y[1] + y[2] + y[3] + g
Subject To
  R0: -1.0 y[0] + -1.0 y[1] + -1.0 y[2] + -1.0 y[3] + -1.0 g <= -1
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
  y[0] + y[1] + y[2] + y[3] + g
Subject To
  R0: -1.0 y[0] + -1.0 y[1] + -1.0 y[2] + -1.0 y[3] + -1.0 g <= -1
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 1 rows, 5 columns and 5 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.000000e+00   0.000000e+00      0s
       1    1.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  1.000000000e+00
    master problem is optimal.
    opt_obj=1.00
    opt_g=0.00
    opt_y0=1.0
    opt_y1=0.0
    opt_y2=0.0
    opt_y3=0.0
Finish solving master problem.
--------------------------------------------------
Bounds: lb=1.00, ub=1.00
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=-0.00
    opt_u0=1.0
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=1.00, ub=1.00
```

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np
import scipy.sparse as sp

class GurobiLpSolver:

    def __init__(self, c, f, A, B, b):
        self._env = gp.Env('GurobiEnv')
        self._model = gp.Model(env=self._env, name='GurobiLpSolver')

        # prepare data
        self._obj_coeff = np.concatenate((c, f))
        print(self._obj_coeff)
        self._constr_mat = np.concatenate((A, B), axis=1)
        print(self._constr_mat)
        self._rhs = b
        self._num_vars = len(self._obj_coeff)
        self._num_constrs = len(b)

        # create decision variables
        self._vars = self._model.addMVar(self._num_vars, vtype=GRB.CONTINUOUS, lb=0)

        # create constraints
        self._constrs = self._model.addConstr(self._constr_mat@self._vars == self._rhs)
        # create objective
        self._model.setObjective(self._obj_coeff @ self._vars, GRB.MINIMIZE)

    def optimize(self):
        self._model.update()
        print(self._model.display())
        self._model.optimize()
        pass

    def clean_up(self):

        pass
```

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np

np.random.seed(142)
c = np.random.randint(2, 6, size=20)
f = np.random.randint(1, 15, size=10)
A = np.random.randint(2, 6, size=(20, 20))
B = np.random.randint(2, 26, size=(20, 10))
b = np.random.randint(20, 50, size=20)

model = GurobiLpSolver(c, f, A, B, b)
model.optimize()
```

```
Set parameter Username
Set parameter LogFile to value "GurobiEnv"
[ 3  3  5  5  2  4  3  2  3  3  4  2  5  5  2  5  4  4  5  5  3  6  8 12
  3  5  4  4  9  3]
[[ 2  4  3  5  5  2  5  4  5  3  3  4  2  2  2  2  3  2  3  3 11 11 25 20
   9  5  7  8  9  2]
 [ 5  2  2  4  3  5  4  2  3  5  4  3  2  5  2  3  5  4  5  4 23 19 23 19
  12 12 13 15 19 19]
 [ 2  3  2  2  5  5  4  5  5  2  5  2  5  4  5  4  3  5  3  3 11 10  8 20
  17  9 25 25 10 12]
 [ 3  5  4  3  4  2  4  2  4  2  4  2  3  3  2  5  3  2  4  2 16 23 18 20
  20 25 14 10  2  6]
 [ 5  4  3  4  3  4  5  4  4  2  5  4  2  3  2  2  5  2  4  3  3  8 10  4
  22 25 11 15 15  9]
 [ 5  5  5  3  2  5  2  2  3  4  2  5  4  4  2  5  4  5  5  4 21 14 22 19
  15 19 16  8 22 23]
 [ 4  2  2  5  2  5  5  4  2  3  2  3  2  5  4  3  3  4  5  3 18  8 11  2
  19 23 23  8 18 25]
 [ 3  2  4  5  3  2  3  5  3  4  5  2  5  4  2  4  2  4  3  5 20 19 13 24
  19  7  4 15 24  3]
 [ 4  4  2  3  2  2  5  5  2  3  5  5  4  5  3  4  2  2  4  2 19  6 20 16
   5 14 20 18 19  6]
 [ 4  2  3  4  4  4  2  5  2  5  5  2  3  4  4  5  4  4  2  3 25 11 17 14
  15 12  6 23 24  6]
 [ 3  5  2  5  3  2  3  3  5  5  3  2  5  3  4  3  5  5  4  4  3 23  9 16
  22  3 14 16 12 16]
 [ 2  3  4  3  5  5  3  5  4  4  2  5  4  3  5  2  3  4  5  4 13 11  7  2
  15 24 13 16  4  6]
 [ 4  4  2  2  4  4  3  2  2  2  5  5  2  5  3  3  5  2  4  3  9 13 10 16
  25 16 24 21  6 16]
 [ 3  5  5  4  3  2  3  5  2  3  5  3  5  3  3  3  5  5  5  5  3 20  5 18
   8 19 25  2 17  7]
 [ 3  2  5  2  2  3  3  3  4  3  5  5  4  4  4  3  3  4  4  4  9 15  6 23
  14 21 21 22 18 22]
 [ 3  2  2  3  4  3  2  4  5  3  2  2  2  3  5  3  2  2  4  4 19 17 17 17
   4 22 12 18  4 18]
 [ 4  2  2  5  2  3  5  4  2  3  2  5  5  5  5  4  4  4  2  5  2 13 23  8
  22 17 10 24 20 23]
 [ 2  4  2  3  2  2  2  4  4  5  2  4  4  4  2  4  3  4  5  3 10  6 20 21
  16 15 13  3 24 16]
 [ 2  5  4  3  3  3  4  2  5  4  2  4  5  2  2  2  3  5  4  5 12 18 21  8
  19  8 11 20 21 21]
 [ 4  2  2  2  2  2  4  3  4  3  2  2  4  2  4  3  5  2  4  3 11  4  2  9
  13 20 23 16 16 14]]
Minimize
3.0 C0 + 3.0 C1 + 5.0 C2 + 5.0 C3 + 2.0 C4 + 4.0 C5 + 3.0 C6 + 2.0 C7 + 3.0 C8
+ 3.0 C9 + 4.0 C10 + 2.0 C11 + 5.0 C12 + 5.0 C13 + 2.0 C14 + 5.0 C15 + 4.0 C16 + 4.0 C17
+ 5.0 C18 + 5.0 C19 + 3.0 C20 + 6.0 C21 + 8.0 C22 + 12.0 C23 + 3.0 C24 + 5.0 C25
+ 4.0 C26 + 4.0 C27 + 9.0 C28 + 3.0 C29
Subject To
R0: 2.0 C0 + 4.0 C1 + 3.0 C2 + 5.0 C3 + 5.0 C4 + 2.0 C5 + 5.0 C6 + 4.0 C7 + 5.0 C8 +
3.0 C9 + 3.0 C10 + 4.0 C11 + 2.0 C12 + 2.0 C13 + 2.0 C14 + 2.0 C15 + 3.0 C16 + 2.0 C17 +
3.0 C18 + 3.0 C19 + 11.0 C20 + 11.0 C21 + 25.0 C22 + 20.0 C23 + 9.0 C24 + 5.0 C25 + 7.0
 C26 + 8.0 C27 + 9.0 C28 + 2.0 C29 = 21
R1: 5.0 C0 + 2.0 C1 + 2.0 C2 + 4.0 C3 + 3.0 C4 + 5.0 C5 + 4.0 C6 + 2.0 C7 + 3.0 C8 +
5.0 C9 + 4.0 C10 + 3.0 C11 + 2.0 C12 + 5.0 C13 + 2.0 C14 + 3.0 C15 + 5.0 C16 + 4.0 C17 +
5.0 C18 + 4.0 C19 + 23.0 C20 + 19.0 C21 + 23.0 C22 + 19.0 C23 + 12.0 C24 + 12.0 C25 +
 13.0 C26 + 15.0 C27 + 19.0 C28 + 19.0 C29 = 26
R2: 2.0 C0 + 3.0 C1 + 2.0 C2 + 2.0 C3 + 5.0 C4 + 5.0 C5 + 4.0 C6 + 5.0 C7 + 5.0 C8 +
2.0 C9 + 5.0 C10 + 2.0 C11 + 5.0 C12 + 4.0 C13 + 5.0 C14 + 4.0 C15 + 3.0 C16 + 5.0 C17 +
3.0 C18 + 3.0 C19 + 11.0 C20 + 10.0 C21 + 8.0 C22 + 20.0 C23 + 17.0 C24 + 9.0 C25 + 25.0
 C26 + 25.0 C27 + 10.0 C28 + 12.0 C29 = 38
R3: 3.0 C0 + 5.0 C1 + 4.0 C2 + 3.0 C3 + 4.0 C4 + 2.0 C5 + 4.0 C6 + 2.0 C7 + 4.0 C8 +
2.0 C9 + 4.0 C10 + 2.0 C11 + 3.0 C12 + 3.0 C13 + 2.0 C14 + 5.0 C15 + 3.0 C16 + 2.0 C17 +
4.0 C18 + 2.0 C19 + 16.0 C20 + 23.0 C21 + 18.0 C22 + 20.0 C23 + 20.0 C24 + 25.0 C25 +
 14.0 C26 + 10.0 C27 + 2.0 C28 + 6.0 C29 = 42
R4: 5.0 C0 + 4.0 C1 + 3.0 C2 + 4.0 C3 + 3.0 C4 + 4.0 C5 + 5.0 C6 + 4.0 C7 + 4.0 C8 +
2.0 C9 + 5.0 C10 + 4.0 C11 + 2.0 C12 + 3.0 C13 + 2.0 C14 + 2.0 C15 + 5.0 C16 + 2.0 C17 +
4.0 C18 + 3.0 C19 + 3.0 C20 + 8.0 C21 + 10.0 C22 + 4.0 C23 + 22.0 C24 + 25.0 C25 + 11.0
 C26 + 15.0 C27 + 15.0 C28 + 9.0 C29 = 35
R5: 5.0 C0 + 5.0 C1 + 5.0 C2 + 3.0 C3 + 2.0 C4 + 5.0 C5 + 2.0 C6 + 2.0 C7 + 3.0 C8 +
4.0 C9 + 2.0 C10 + 5.0 C11 + 4.0 C12 + 4.0 C13 + 2.0 C14 + 5.0 C15 + 4.0 C16 + 5.0 C17 +
5.0 C18 + 4.0 C19 + 21.0 C20 + 14.0 C21 + 22.0 C22 + 19.0 C23 + 15.0 C24 + 19.0 C25 +
 16.0 C26 + 8.0 C27 + 22.0 C28 + 23.0 C29 = 37
R6: 4.0 C0 + 2.0 C1 + 2.0 C2 + 5.0 C3 + 2.0 C4 + 5.0 C5 + 5.0 C6 + 4.0 C7 + 2.0 C8 +
3.0 C9 + 2.0 C10 + 3.0 C11 + 2.0 C12 + 5.0 C13 + 4.0 C14 + 3.0 C15 + 3.0 C16 + 4.0 C17 +
5.0 C18 + 3.0 C19 + 18.0 C20 + 8.0 C21 + 11.0 C22 + 2.0 C23 + 19.0 C24 + 23.0 C25 + 23.0
 C26 + 8.0 C27 + 18.0 C28 + 25.0 C29 = 28
R7: 3.0 C0 + 2.0 C1 + 4.0 C2 + 5.0 C3 + 3.0 C4 + 2.0 C5 + 3.0 C6 + 5.0 C7 + 3.0 C8 +
4.0 C9 + 5.0 C10 + 2.0 C11 + 5.0 C12 + 4.0 C13 + 2.0 C14 + 4.0 C15 + 2.0 C16 + 4.0 C17 +
3.0 C18 + 5.0 C19 + 20.0 C20 + 19.0 C21 + 13.0 C22 + 24.0 C23 + 19.0 C24 + 7.0 C25 + 4.0
 C26 + 15.0 C27 + 24.0 C28 + 3.0 C29 = 22
R8: 4.0 C0 + 4.0 C1 + 2.0 C2 + 3.0 C3 + 2.0 C4 + 2.0 C5 + 5.0 C6 + 5.0 C7 + 2.0 C8 +
3.0 C9 + 5.0 C10 + 5.0 C11 + 4.0 C12 + 5.0 C13 + 3.0 C14 + 4.0 C15 + 2.0 C16 + 2.0 C17 +
4.0 C18 + 2.0 C19 + 19.0 C20 + 6.0 C21 + 20.0 C22 + 16.0 C23 + 5.0 C24 + 14.0 C25 + 20.0
 C26 + 18.0 C27 + 19.0 C28 + 6.0 C29 = 39
R9: 4.0 C0 + 2.0 C1 + 3.0 C2 + 4.0 C3 + 4.0 C4 + 4.0 C5 + 2.0 C6 + 5.0 C7 + 2.0 C8 +
5.0 C9 + 5.0 C10 + 2.0 C11 + 3.0 C12 + 4.0 C13 + 4.0 C14 + 5.0 C15 + 4.0 C16 + 4.0 C17 +
2.0 C18 + 3.0 C19 + 25.0 C20 + 11.0 C21 + 17.0 C22 + 14.0 C23 + 15.0 C24 + 12.0 C25 +
 6.0 C26 + 23.0 C27 + 24.0 C28 + 6.0 C29 = 33
R10: 3.0 C0 + 5.0 C1 + 2.0 C2 + 5.0 C3 + 3.0 C4 + 2.0 C5 + 3.0 C6 + 3.0 C7 + 5.0 C8 +
5.0 C9 + 3.0 C10 + 2.0 C11 + 5.0 C12 + 3.0 C13 + 4.0 C14 + 3.0 C15 + 5.0 C16 + 5.0 C17 +
4.0 C18 + 4.0 C19 + 3.0 C20 + 23.0 C21 + 9.0 C22 + 16.0 C23 + 22.0 C24 + 3.0 C25 + 14.0
 C26 + 16.0 C27 + 12.0 C28 + 16.0 C29 = 28
R11: 2.0 C0 + 3.0 C1 + 4.0 C2 + 3.0 C3 + 5.0 C4 + 5.0 C5 + 3.0 C6 + 5.0 C7 + 4.0 C8 +
4.0 C9 + 2.0 C10 + 5.0 C11 + 4.0 C12 + 3.0 C13 + 5.0 C14 + 2.0 C15 + 3.0 C16 + 4.0 C17 +
5.0 C18 + 4.0 C19 + 13.0 C20 + 11.0 C21 + 7.0 C22 + 2.0 C23 + 15.0 C24 + 24.0 C25 + 13.0
 C26 + 16.0 C27 + 4.0 C28 + 6.0 C29 = 38
R12: 4.0 C0 + 4.0 C1 + 2.0 C2 + 2.0 C3 + 4.0 C4 + 4.0 C5 + 3.0 C6 + 2.0 C7 + 2.0 C8 +
2.0 C9 + 5.0 C10 + 5.0 C11 + 2.0 C12 + 5.0 C13 + 3.0 C14 + 3.0 C15 + 5.0 C16 + 2.0 C17 +
4.0 C18 + 3.0 C19 + 9.0 C20 + 13.0 C21 + 10.0 C22 + 16.0 C23 + 25.0 C24 + 16.0 C25 +
 24.0 C26 + 21.0 C27 + 6.0 C28 + 16.0 C29 = 47
R13: 3.0 C0 + 5.0 C1 + 5.0 C2 + 4.0 C3 + 3.0 C4 + 2.0 C5 + 3.0 C6 + 5.0 C7 + 2.0 C8 +
3.0 C9 + 5.0 C10 + 3.0 C11 + 5.0 C12 + 3.0 C13 + 3.0 C14 + 3.0 C15 + 5.0 C16 + 5.0 C17 +
5.0 C18 + 5.0 C19 + 3.0 C20 + 20.0 C21 + 5.0 C22 + 18.0 C23 + 8.0 C24 + 19.0 C25 + 25.0
 C26 + 2.0 C27 + 17.0 C28 + 7.0 C29 = 28
R14: 3.0 C0 + 2.0 C1 + 5.0 C2 + 2.0 C3 + 2.0 C4 + 3.0 C5 + 3.0 C6 + 3.0 C7 + 4.0 C8 +
3.0 C9 + 5.0 C10 + 5.0 C11 + 4.0 C12 + 4.0 C13 + 4.0 C14 + 3.0 C15 + 3.0 C16 + 4.0 C17 +
4.0 C18 + 4.0 C19 + 9.0 C20 + 15.0 C21 + 6.0 C22 + 23.0 C23 + 14.0 C24 + 21.0 C25 + 21.0
 C26 + 22.0 C27 + 18.0 C28 + 22.0 C29 = 45
R15: 3.0 C0 + 2.0 C1 + 2.0 C2 + 3.0 C3 + 4.0 C4 + 3.0 C5 + 2.0 C6 + 4.0 C7 + 5.0 C8 +
3.0 C9 + 2.0 C10 + 2.0 C11 + 2.0 C12 + 3.0 C13 + 5.0 C14 + 3.0 C15 + 2.0 C16 + 2.0 C17 +
4.0 C18 + 4.0 C19 + 19.0 C20 + 17.0 C21 + 17.0 C22 + 17.0 C23 + 4.0 C24 + 22.0 C25 +
 12.0 C26 + 18.0 C27 + 4.0 C28 + 18.0 C29 = 34
R16: 4.0 C0 + 2.0 C1 + 2.0 C2 + 5.0 C3 + 2.0 C4 + 3.0 C5 + 5.0 C6 + 4.0 C7 + 2.0 C8 +
3.0 C9 + 2.0 C10 + 5.0 C11 + 5.0 C12 + 5.0 C13 + 5.0 C14 + 4.0 C15 + 4.0 C16 + 4.0 C17 +
2.0 C18 + 5.0 C19 + 2.0 C20 + 13.0 C21 + 23.0 C22 + 8.0 C23 + 22.0 C24 + 17.0 C25 + 10.0
 C26 + 24.0 C27 + 20.0 C28 + 23.0 C29 = 46
R17: 2.0 C0 + 4.0 C1 + 2.0 C2 + 3.0 C3 + 2.0 C4 + 2.0 C5 + 2.0 C6 + 4.0 C7 + 4.0 C8 +
5.0 C9 + 2.0 C10 + 4.0 C11 + 4.0 C12 + 4.0 C13 + 2.0 C14 + 4.0 C15 + 3.0 C16 + 4.0 C17 +
5.0 C18 + 3.0 C19 + 10.0 C20 + 6.0 C21 + 20.0 C22 + 21.0 C23 + 16.0 C24 + 15.0 C25 +
 13.0 C26 + 3.0 C27 + 24.0 C28 + 16.0 C29 = 35
R18: 2.0 C0 + 5.0 C1 + 4.0 C2 + 3.0 C3 + 3.0 C4 + 3.0 C5 + 4.0 C6 + 2.0 C7 + 5.0 C8 +
4.0 C9 + 2.0 C10 + 4.0 C11 + 5.0 C12 + 2.0 C13 + 2.0 C14 + 2.0 C15 + 3.0 C16 + 5.0 C17 +
4.0 C18 + 5.0 C19 + 12.0 C20 + 18.0 C21 + 21.0 C22 + 8.0 C23 + 19.0 C24 + 8.0 C25 + 11.0
 C26 + 20.0 C27 + 21.0 C28 + 21.0 C29 = 46
R19: 4.0 C0 + 2.0 C1 + 2.0 C2 + 2.0 C3 + 2.0 C4 + 2.0 C5 + 4.0 C6 + 3.0 C7 + 4.0 C8 +
3.0 C9 + 2.0 C10 + 2.0 C11 + 4.0 C12 + 2.0 C13 + 4.0 C14 + 3.0 C15 + 5.0 C16 + 2.0 C17 +
4.0 C18 + 3.0 C19 + 11.0 C20 + 4.0 C21 + 2.0 C22 + 9.0 C23 + 13.0 C24 + 20.0 C25 + 23.0
 C26 + 16.0 C27 + 16.0 C28 + 14.0 C29 = 32
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 20 rows, 30 columns and 600 nonzeros
Model fingerprint: 0x05cb70d4
Coefficient statistics:
  Matrix range     [2e+00, 2e+01]
  Objective range  [2e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 5e+01]
```

```
Presolve time: 0.00s
Presolved: 20 rows, 30 columns, 600 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.3913043e+00   5.119565e+01   0.000000e+00      0s

Solved in 13 iterations and 0.01 seconds (0.00 work units)
Infeasible model
```

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np

np.random.seed(142)
c = np.random.randint(2, 6, size=20)
f = np.random.randint(1, 15, size=10)
A = np.random.randint(2, 6, size=(20, 20))
B = np.random.randint(2, 26, size=(20, 10))
b = np.random.randint(20, 50, size=20)

master_solver = GenericLpMasterSolver(f, B, b)
dual_subprob_solver = GenericLpSubprobSolver(A, c, B, b)

benders_solver = GenericBendersSolver(master_solver, dual_subprob_solver)
benders_solver.optimize()
```

```
Set parameter Username
Set parameter LogFile to value "MasterEnv"
Set parameter Username
Set parameter LogFile to value "SubprobEnv"
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads
```

```
Optimize a model with 0 rows, 11 columns and 0 nonzeros
Model fingerprint: 0xb01aa8a2
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 0 rows and 11 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  0.000000000e+00
    master problem is optimal.
    opt_obj=0.00
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=0.00, ub=inf
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=28.16
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=0.00, ub=28.16
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
```

```
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 1 rows, 11 columns and 11 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.759766e+00   0.000000e+00      0s
       1    6.9664948e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  6.966494845e+00
    master problem is optimal.
    opt_obj=6.97
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=6.97, ub=28.16
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=11.82
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=6.97, ub=18.78
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
```

```
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 2 rows, 11 columns and 22 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    6.9664948e+00   7.385755e-01   0.000000e+00      0s
       1    7.9710765e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  7.971076459e+00
    master problem is optimal.
    opt_obj=7.97
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=7.97, ub=18.78
```

```
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=14.74
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=7.97, ub=18.78
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])
```

```
CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 11 columns and 33 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.9710765e+00   9.214938e-01   0.000000e+00      0s
       1    8.3297685e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  8.329768548e+00
    master problem is optimal.
    opt_obj=8.33
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=8.33, ub=18.78
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=5.92
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=8.33, ub=14.25
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
```

```
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 4 rows, 11 columns and 44 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    8.3297685e+00   3.701343e-01   0.000000e+00      0s
       1    8.9196232e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  8.919623169e+00
    master problem is optimal.
    opt_obj=8.92
    opt_g=0.00
```

```
Finish solving master problem.
--------------------------------------------------
Bounds: lb=8.92, ub=14.25
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=6.12
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=8.92, ub=14.25
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
```

```
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 5 rows, 11 columns and 55 nonzeros
Coefficient statistics:
  Matrix range     [8e-01, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    8.9196232e+00   3.825374e-01   0.000000e+00      0s
       1    9.5016530e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  9.501652960e+00
    master problem is optimal.
    opt_obj=9.50
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=9.50, ub=14.25
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=2.23
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=9.50, ub=11.73
Minimize
```

```
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 6 rows, 11 columns and 66 nonzeros
Coefficient statistics:
  Matrix range     [8e-01, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.5016530e+00   2.782364e-01   0.000000e+00      0s
       1    9.6700817e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.01 seconds (0.00 work units)
Optimal objective  9.670081743e+00
    master problem is optimal.
    opt_obj=9.67
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=9.67, ub=11.73
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=2.23
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=9.67, ub=11.73
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
```

```
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
R6: -5.0 y[0] + -3.0 y[1] + -10.0 y[2] + -10.5 y[3] + -8.0 y[4] + -7.5 y[5] + -6.5 y[6]
 + -1.5 y[7] + -12.0 y[8] + -8.0 y[9] + -1.0 g <= -17.5
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
```

```
R6: -5.0 y[0] + -3.0 y[1] + -10.0 y[2] + -10.5 y[3] + -8.0 y[4] + -7.5 y[5] + -6.5 y[6]
 + -1.5 y[7] + -12.0 y[8] + -8.0 y[9] + -1.0 g <= -17.5
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 7 rows, 11 columns and 77 nonzeros
Coefficient statistics:
  Matrix range     [8e-01, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.6700817e+00   1.396581e-01   0.000000e+00      0s
       1    9.8201416e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  9.820141589e+00
    master problem is optimal.
    opt_obj=9.82
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=9.82, ub=11.73
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=1.40
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=9.82, ub=11.22
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
```

```
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
R6: -5.0 y[0] + -3.0 y[1] + -10.0 y[2] + -10.5 y[3] + -8.0 y[4] + -7.5 y[5] + -6.5 y[6]
 + -1.5 y[7] + -12.0 y[8] + -8.0 y[9] + -1.0 g <= -17.5
R7: -1.2000000000000002 y[0] + -8.0 y[1] + -2.0 y[2] + -7.2 y[3] + -3.2 y[4] +
-7.6000000000000005 y[5] + -10.0 y[6] + -0.8 y[7] + -6.800000000000001 y[8] +
 -2.8000000000000003 y[9] + -1.0 g <= -11.2
None
Benders optimality cut added!
--------------------------------------------------
Start solving master problem.
Minimize
3.0 y[0] + 6.0 y[1] + 8.0 y[2] + 12.0 y[3] + 3.0 y[4] + 5.0 y[5] + 4.0 y[6] + 4.0 y[7]
+ 9.0 y[8] + 3.0 y[9] + g
Subject To
R0: -8.5625 y[0] + -9.859375 y[1] + -9.5625 y[2] + -7.9375 y[3] + -12.125 y[4] +
-11.828125 y[5] + -11.21875 y[6] + -11.46875 y[7] + -10.21875 y[8] + -10.5 y[9] + -1.0 g
 <= -28.1562
R1: -8.76923076923077 y[0] + -6.153846153846153 y[1] + -8.307692307692307 y[2] +
-7.692307692307692 y[3] + -2.0 y[4] + -8.923076923076923 y[5] + -6.769230769230769 y[6]
+ -8.307692307692307 y[7] + -4.153846153846153 y[8] + -6.461538461538462 y[9] + -1.0 g
 <= -16.4615
R2: -10.75 y[0] + -10.124999999999998 y[1] + -11.75 y[2] + -11.5 y[3] + -8.75 y[4] +
-12.874999999999998 y[5] + -10.25 y[6] + -8.25 y[7] + -5.5 y[8] + -3.7499999999999996
 y[9] + -1.0 g <= -25.5
R3: -3.0 y[0] + -8.75 y[1] + -2.75 y[2] + -10.25 y[3] + -5.5 y[4] + -10.0 y[5] + -11.5
 y[6] + -6.0 y[7] + -8.75 y[8] + -7.25 y[9] + -1.0 g <= -18.25
R4: -0.8 y[0] + -5.2 y[1] + -9.200000000000001 y[2] + -3.2 y[3] + -8.8 y[4] +
-6.800000000000001 y[5] + -4.0 y[6] + -9.600000000000001 y[7] + -8.0 y[8] +
 -9.200000000000001 y[9] + -1.0 g <= -18.4
R5: -7.6000000000000005 y[0] + -2.4000000000000004 y[1] + -8.0 y[2] + -6.4 y[3] + -2.0
y[4] + -5.6000000000000005 y[5] + -8.0 y[6] + -7.2 y[7] + -7.6000000000000005 y[8] +
 -2.4000000000000004 y[9] + -1.0 g <= -15.6
R6: -5.0 y[0] + -3.0 y[1] + -10.0 y[2] + -10.5 y[3] + -8.0 y[4] + -7.5 y[5] + -6.5 y[6]
 + -1.5 y[7] + -12.0 y[8] + -8.0 y[9] + -1.0 g <= -17.5
R7: -1.2000000000000002 y[0] + -8.0 y[1] + -2.0 y[2] + -7.2 y[3] + -3.2 y[4] +
-7.6000000000000005 y[5] + -10.0 y[6] + -0.8 y[7] + -6.800000000000001 y[8] +
 -2.8000000000000003 y[9] + -1.0 g <= -11.2
None
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 8 rows, 11 columns and 88 nonzeros
Coefficient statistics:
  Matrix range     [8e-01, 1e+01]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 3e+01]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.8201416e+00   8.763627e-02   0.000000e+00      0s
       1    9.8617265e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Optimal objective  9.861726459e+00
    master problem is optimal.
    opt_obj=9.86
    opt_g=0.00
Finish solving master problem.
--------------------------------------------------
Bounds: lb=9.86, ub=11.22
dual subproblem objective updated!
--------------------------------------------------
Start solving dual subproblem.
    dual subproblem is optimal.
    opt_obj=0.00
Finish solving dual subproblem.
--------------------------------------------------
Bounds: lb=9.86, ub=9.86
```

```
c
```

```
array([4, 5, 2, 4, 4, 5, 2, 2, 4, 3, 4, 4, 4, 4, 5, 2, 5, 5, 5, 4])
```

```
np.random.randint(1, 5, size=(2, 10))
```

```
array([[4, 1, 4, 2, 2, 3, 2, 1, 4, 4],
       [4, 4, 2, 1, 1, 2, 2, 3, 4, 3]])
```

4.2.7 Implementation with callbacks

4.2.8 Implementation with SCIP

```
from pyscipopt import Model
from pyscipopt import quicksum
from pyscipopt import SCIP_PARAMSETTING

# Create a model
model = Model("simple_lp")

# Define variables
x1 = model.addVar(lb=0, vtype="C", name="x1")
x2 = model.addVar(lb=0, vtype="C", name="x2")

# Set objective function
model.setObjective(x1 + x2, "maximize")

# Add constraints
# model.addCons(2 * x1 + x2 >= 1, "constraint1")
model.addCons(x1 + x2 >= 2, "constraint2")

# Solve the model
model.setPresolve(SCIP_PARAMSETTING.OFF)
model.setHeuristics(SCIP_PARAMSETTING.OFF)
model.disablePropagation()
model.optimize()

# Print results
status = model.getStatus()
print(f'status = {status}')
if model.getStatus() == "optimal":
    print("Optimal solution found.")
    print(f"x1: {model.getVal(x1):.2f}")
    print(f"x2: {model.getVal(x2):.2f}")
    print(f"Objective value: {model.getObjVal():.2f}")
    hasRay = model.hasPrimaryRay()
    print(hasRay)
elif model.getStatus() == 'unbounded':
    hasRay = model.hasPrimalRay()
    print(f'hasRay={hasRay}')
    ray = model.getPrimalRay()
    print(f'ray={ray}')

else:
    print("Model could not be solved.")
```

```
status = unbounded
hasRay=True
ray=[0.5, 0.5]
presolving:
   (0.0s) symmetry computation started: requiring (bin +, int +, cont +), (fixed: bin -, int
   (0.0s) symmetry computation finished: 1 generators found (max: 1500, log10 of symmetry gro
   (0.0s) no symmetry on binary variables present.
presolving (0 rounds: 0 fast, 0 medium, 0 exhaustive):
 0 deleted vars, 0 deleted constraints, 0 added constraints, 0 tightened bounds, 0 added hole
 0 implications, 0 cliques
presolved problem has 2 variables (0 bin, 0 int, 0 impl, 2 cont) and 2 constraints
      2 constraints of type <linear>
Presolving Time: 0.00

 time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|strb
* 0.0s|     1 |     0 |     2 |     - |    LP  |   0 |   2 |   2 |   2 |   1 |  0 |   0 |   0

SCIP Status        : problem is solved [unbounded]
Solving Time (sec) : 0.00
Solving Nodes      : 1
Primal Bound       : +1.00000000000000e+20 (1 solutions)
Dual Bound         : +1.00000000000000e+20
Gap                : 0.00 %
```

Testing

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Listing 4.1 A LP solver based on Gurobi

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np

class LpSolverGurobi:

    def __init__(self, obj_coeff, constr_mat, rhs):
        # initialize environment and model
        self._env = gp.Env('GurobiEnv')
        self._model = gp.Model(env=self._env, name='GurobiLpSolver')

        # prepare data
        self._obj_coeff = obj_coeff
        # print(self._obj_coeff)
        self._constr_mat = constr_mat
        # print(self._constr_mat)
        self._rhs = rhs
        self._num_vars = len(self._obj_coeff)
        self._num_constrs = len(self._rhs)

        # create decision variables
        self._vars = self._model.addMVar(self._num_vars,
                                         vtype=GRB.CONTINUOUS,
                                         lb=0)

        # create constraints
        self._constrs = self._model.addConstr(
            self._constr_mat@self._vars == self._rhs
        )

        # create objective
        self._model.setObjective(self._obj_coeff@self._vars,
                                 GRB.MINIMIZE)

    def optimize(self):
        self._model.optimize()
        if self._model.status == GRB.OPTIMAL:
            print(f'Optimal solution found!')
            print(f'Optimal objective = {self._model.objVal:.2f}')
        elif self._model.status == GRB.UNBOUNDED:
            print(f'Model is unbounded!')
        elif self._model.status == GRB.INFEASIBLE:
            print(f'Model is infeasible!')
        else:
            print(f'Unknown error occurred!')

    def clean_up(self):
        self._model.dispose()
        self._env.dispose()
```

Listing 4.2 A LP solver based on SCIP

```
import pyscipopt as scip
from pyscipopt import SCIP_PARAMSETTING

class LpSolverSCIP:

    def __init__(self, obj_coeff, constr_mat, rhs):
        self._model = scip.Model('LpModel')

        # create variables
        self._vars = {
            i: self._model.addVar(lb=0, vtype='C')
            for i in range(len(obj_coeff))
        }

        # create constraints
        for c in range(len(rhs)):
            expr = [
                constr_mat[c][j] * self._vars.get(j)
                for j in range(len(obj_coeff))
            ]
            self._model.addCons(scip.quicksum(expr) == rhs[c])

        # create objective
        obj_expr = [
            obj_coeff[i] * self._vars.get(i)
            for i in range(len(obj_coeff))
        ]
        self._model.setObjective(scip.quicksum(obj_expr), "minimize")

    def optimize(self):
        self._model.optimize()
        status = self._model.getStatus()
        if status == "optimal":
            print(f'Optimal solution found!')
            print(f'Optimal objective = {self._model.getObjVal():.2f}')
        elif self._model.status == "unbounded":
            print(f'Model is unbounded!')
        elif self._model.status == "infeasible":
            print(f'Model is infeasible!')
        else:
            print(f'Unknown error occurred!')
```

Listing 4.3 A randomly generated LP problem

```
import numpy as np

np.random.seed(42)
c = np.random.randint(1, 6, size=20)
A = np.random.randint(-10, 12, size=(5, 20))
b = np.random.randint(20, 100, size=5)
```

Listing 4.4 Solving the generated LP with Gurobi

```
lpsolver_gurobi = LpSolverGurobi(obj_coeff=c, constr_mat=A, rhs=b)
lpsolver_gurobi.optimize()
```

Listing 4.5 Solver outputs

```
# Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (mac64[arm])

# CPU model: Apple M1
# Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

# Optimize a model with 5 rows, 20 columns and 99 nonzeros
# Model fingerprint: 0x472c52fb
# Coefficient statistics:
#   Matrix range     [1e+00, 1e+01]
#   Objective range  [1e+00, 5e+00]
#   Bounds range     [0e+00, 0e+00]
#   RHS range        [2e+01, 5e+01]
# Presolve time: 0.00s
# Presolved: 5 rows, 20 columns, 99 nonzeros

# Iteration    Objective       Primal Inf.    Dual Inf.      Time
#        0    0.0000000e+00   1.887500e+01   0.000000e+00      0s
#        5    3.6899639e+01   0.000000e+00   0.000000e+00      0s

# Solved in 5 iterations and 0.00 seconds (0.00 work units)
# Optimal objective  3.689963907e+01
# Optimal solution found!
# Optimal objective = 36.90
```

Listing 4.6 Solving the generated LP with SCIP

```
lpsolver_scip = LpSolverSCIP(obj_coeff=c, constr_mat=A, rhs=b)
lpsolver_scip.optimize()
```

Listing 4.7 Solver outputs

```
# presolving:
#    (0.0s) running MILP presolver
#    (0.0s) MILP presolver found nothing
#    (0.0s) sparsify finished: 10/99 (10.1%) nonzeros canceled - in total 10 canceled nonzero
# (round 1, exhaustive) 0 del vars, 0 del conss, 0 add conss, 0 chg bounds, 0 chg sides, 77 
#    (0.0s) symmetry computation started: requiring (bin +, int +, cont +), (fixed: bin -, i
#    (0.0s) no symmetry present (symcode time: 0.00)
# presolving (2 rounds: 2 fast, 2 medium, 2 exhaustive):
#  0 deleted vars, 0 deleted constraints, 0 added constraints, 0 tightened bounds, 0 added h
#  0 implications, 0 cliques
# presolved problem has 20 variables (0 bin, 0 int, 0 impl, 20 cont) and 5 constraints
#      5 constraints of type <linear>
# Presolving Time: 0.00

#  time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|s
# * 0.0s|     1 |     0 |     7 |     - |    LP  |   0 |  20 |   5 |   5 |   0 |  0 |   0 |
#   0.0s|     1 |     0 |     7 |     - |   769k |   0 |  20 |   5 |   5 |   0 |  0 |   0 |

# SCIP Status        : problem is solved [optimal solution found]
# Solving Time (sec) : 0.00
# Solving Nodes      : 1
# Primal Bound       : +3.68996390687687e+01 (1 solutions)
# Dual Bound         : +3.68996390687687e+01
# Gap                : 0.00 %

# Optimal objective = 36.90
```