

Module 2

Ryo Kimura

March 21, 2019

2 Optimization Software

In this module we will provide an overview of the main optimization solvers we will be using throughout the course, Gurobi and CPLEX; where they fit in the class of optimization solvers more generally, what problems they can solve, and how to install them.

2.1 Problem Classes and Solvers

One way to classify some of the optimization models used in operations research is by the types of constraints and objective function you are allowed to use. Since these differences often necessitate different solution algorithms, the classification also tends to correspond to different solvers.

The solvers we will be working with most extensively (i.e., Gurobi and CPLEX) are *MIP solvers*, which are designed to solve the following types of problems:

- **Linear Programming (LP)**: linear objective functions, affine inequality constraints
- **(Mixed) Integer Linear Programming (ILP, MILP)**: linear objective functions, affine inequality constraints, integrality constraints
- **Quadratic Programming (QP, QCP, SOCP)**: quadratic objective functions, convex quadratic inequality constraints
- **Mixed Integer Quadratic Programming (MIQP)**: quadratic objective functions, convex quadratic inequality constraints, integrality constraints

Other classes of optimization problems that are also solvable in practice include:

- **Conic Optimization (SDP, SCP)**: linear objective functions, (convex) conic constraints; popular solvers include Mosek and CVX
- **Mixed Integer Nonlinear Programming**: nonlinear (nonconvex) objective functions and constraints, integrality constraints; popular solvers include BARON and LINGO
- **Constraint Programming (CP)**: general objective functions, logical constraints, finite domain variables; popular solvers include CP Optimizer, Gecode, and Choco
- **Combinatorial Optimization (TSP, Matching, etc.)**: structured objective and feasible solutions; popular solvers include Concorde (TSP) and NetworkX (graph problems)


2.2 Gurobi

Gurobi is a MIP solver that was first introduced in 2008 by three former developers of CPLEX: Zhonghao **Gu**, Edward **Rothberg**, and Robert **Bixby**. It is regarded as one of the best general purpose MIP solvers today, with competitive performance on a wide range of problems and continual improvements to the core solver.

While Gurobi is a commercial solver, it provides academic licenses free of charge. While each license can only be installed for a single physical (or virtual) machine, there is no limit on the number of free licenses you can obtain.

Setting up Gurobi on your machine involves four steps: (1) obtaining an academic license, (2) installing Gurobi on your machine, (3) retrieving the license on your machine, and (4) setting up the API for your programming language. See Gurobi's **Quick Start Guides** (<http://www.gurobi.com/documentation/>) for details.

2.2.1 Obtaining an (Individual) Academic License

1. Go to <http://www.gurobi.com/registration/academic-license-reg>.
2. If you don't already have a Gurobi account, register for one. Log in to your account.
3. Accept the EULA and conditions for an academic license (basically you cannot use it for commercial purposes), and press **Request License**.
4. Your requested license will immediately appear in your account. You can access your current licenses by hovering over , selecting **My Account**, then clicking on **Review Your Current Licenses** under **My Licenses**.

2.2.2 Installing Gurobi

On Linux, perform the following steps (for the steps marked [**Admin**], you will need admin privileges, so either put `sudo` before the command or log in to the `root` account):

1. Go to <http://www.gurobi.com/downloads/gurobi-optimizer> and download the latest **64-bit Linux** version of Gurobi (you will need to accept the EULA). This will be in the form of a *targz archive* (e.g., `gurobiX.Y.Z_linux64.tar.gz`).
2. [**Admin**] If you don't already have an `/opt` directory in your root directory, create one (see section 1.3.1 of Module 1).
3. [**Admin**] Move the targz archive to the `/opt` directory (see section 1.1.2 of Module 1).
4. [**Admin**] Extract the targz archive in the `/opt` directory (see section 1.3.1 of Module 1).
5. Open the `.bashrc` file in your home directory (e.g., `/home/alice/.bashrc`) and add the following lines to modify the environment variables `GUROBI_HOME`, `PATH`, and `LD_LIBRARY_PATH`:

```
export GUROBI_HOME="/opt/gurobiXXX/linux64"
export PATH="${PATH}:${GUROBI_HOME}/bin"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${GUROBI_HOME}/lib"
```

where XXX is replaced with appropriate version numbers. Note that, unlike many programming languages, whitespaces matters in Bash; in particular, make sure there are no spaces on either side of `=`.

6. You may need to restart Bash in order for the modified environment variables to take effect.

On Windows, the installation process is much simpler.

1. Go to <http://www.gurobi.com/downloads/gurobi-optimizer> and download the latest **64-bit Windows** version of Gurobi (you will need to accept the EULA). This will be in the form of an *MSI file* (e.g., `Gurobi-X.Y.Z-win64.msi`).
2. Run the MSI file.

2.2.3 Retrieving the License

1. Go to <https://user.gurobi.com/download/licenses/current> and click on the license.
2. Run the `grbgetkey` command at the bottom of your license page from a command line terminal (on Windows, you can also run it by copying the command into the Run app). Note that you must be on CMU's network in order for `grbgetkey` to work.¹
3. When prompted, save the license in the default location (i.e., your home directory).

2.2.4 Setting up the Python API

Assuming Python is already installed (see Module 1), there are two ways to set up the Gurobi Python API depending on whether you are using Anaconda or not:

- If you are using **Anaconda**, you can install the official Gurobi Anaconda package by opening a terminal and running the commands

```
conda config --add channels gurobi
conda install gurobi
```

You can remove the Gurobi package at any time by running `conda remove gurobi`. See <http://www.gurobi.com/downloads/get-anaconda> for details.

- If you are not using Anaconda, you need to install Gurobi's Python package by directly running `setup.py`.² Follow these steps:

1. Navigate to the root Gurobi directory (usually `/opt/gurobiXXX/linux64/` or `C:\Program Files\GurobiXXX\win64\` where XXX is replaced with appropriate version numbers); it should contain a file called `setup.py`.

¹You can install Gurobi to a machine that is not on CMU's network (e.g., a home desktop) by using a *VPN client* (see section 1.2.2 of Module 1). See https://www.gurobi.com/documentation/current/quickstart_linux/academic_validation.html and <http://www.gurobi.com/support/faqs#AL> for more details.

²While running `setup.py` directly is usually **not** recommended, in this case it is necessary due to Gurobi's use of non-standard features of distutils.

2. Run the command `python3 setup.py install` in the current directory.³⁴

You should now be able to run Gurobi's Python examples. See http://www.gurobi.com/documentation/current/refman/py_python_api_overview.html for details.⁵

2.2.5 Setting up the C++ API

Assuming a C++ compiler is already installed (see Module 1), there is no additional setup needed to use the Gurobi C++ API unless you are using a build system (which we will cover in Module 5). Here are two simple ways to compile Gurobi's C++ examples:

- To compile directly from the command line, run a command similar to⁶

```
g++ -m64 -g -o example example.cpp -I${GUROBI_HOME} -L${GUROBI_HOME}/lib \
    -lgurobi_c++ -lgurobi81 -lm
```

(Note that `\` is simply an indicator that the command is continued on the following line.)

- Alternatively, you can use a *Makefile*, which runs the compilation command for you based on some specified patterns. See the Canvas module for a minimal Makefile based on the ones provided by Gurobi and how to use it. See http://www.gurobi.com/documentation/current/quickstart_linux/cpp_building_and_running_t.html (in the **Quick Start Guides** under **C++ Interface** » **Building and running the example**) for details.

See http://www.gurobi.com/documentation/current/refman/cpp_api_overview.html for details.

2.3 CPLEX

CPLEX is a MIP solver that was first introduced by Robert Bixby in 1988 as an LP solver, and later extended to a MIP solver in 1992. With decades of development and an illustrious history, it is one of the most well-respected MIP solvers that has long served as the de facto standard in the field. CPLEX is part of the larger optimization software package *IBM ILOG CPLEX Optimization Studio*, which also includes the constraint programming solver *CP Optimizer*, the high-level modeling language *OPL (Optimization Programming Language)*, and a tightly integrated development environment called *CPLEX Studio IDE*.

While CPLEX is a commercial MIP solver, it distributes its software free to academics through the *IBM Academic Initiative* program. Unlike Gurobi, CPLEX does not have licenses, but rather they restrict downloads by requiring users to “buy” CPLEX through an online store.

³If you get an error saying `distutils.base` is not found, you are missing the `distutils` Python module. In this case, it is **recommended** that you install `pip/pip3`, which will install `distutils` as a dependency.

⁴If you don't have Admin privileges you can run `python3 setup.py build -b /tmp install --user` instead to install the package to your user site directory. You can run the command `python3 -m site` to see where your user site directory is.

⁵If you have tried the recommended options and you still get import errors, as a last resort you can create a soft link to the directory containing the Python package *in the same directory as the Python script that needs it*. Note that the soft link *must* be named `gurobipy`. You can create soft links in Linux using the command `ln -s`.

⁶Note that depending on the version of GCC you have (use the command `g++ --version` to find out), you may need to replace `-lgurobi_c++` with `-lgurobi_g++5.2` or `-lgurobi_g++4.2`.

Setting up CPLEX on your machine involves three steps: (1) downloading CPLEX, (2) installing CPLEX on your machine, and (3) setting up the API for your programming language. See the following links for details:

- IBM Academic Resources FAQ (<https://developer.ibm.com/academic/frequently-asked-questions/#downloadquestions>)
- **Getting Started with CPLEX** from the CPLEX documentation (https://www.ibm.com/support/knowledgecenter/en/SSSA5P_latest/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/set_up/setup_synopsis.html or navigate **CPLEX** » **Getting Started with CPLEX** » **Setting up CPLEX**)

2.3.1 Downloading CPLEX

1. Go to <https://ibm.onthehub.com>.
2. Register for an OnTheHub account. You will be asked to verify your email. Note that this is completely separate from your IBM id (if you have one).
3. Select CPLEX through the store by clicking **Home Page** and navigating to **Students** » **Data & Analytics** » **Software** » **CPLEX** (this can be a bit tricky). Click **Add to Cart**, then click **Check Out**.
4. Accept the Academic Initiative Program Agreement, and answer the 4 question survey.
5. At this point, your software request may get (erroneously) rejected. Don't panic; usually the error is resolved in about 10-15 minutes. If not, email them about the error at <https://ibm.onthehub.com/WebStore/Support/ContactUs.aspx>.
6. Download the appropriate version of the software (Linux x86-64 or Windows x86-64).

2.3.2 Installing CPLEX

On Linux, perform the following steps (for the steps marked **[Admin]**, you will need admin privileges, so either put `sudo` before the command or log in to the `root` account):

1. Navigate to location of the downloaded binary (see section 1.1.2 of Module 1).
2. **[Admin]** Run the binary (it's a universal shell script with embedded data).
3. Enter 2 to choose the English locale (or a different one if desired).
4. Enter 1 to accept the license agreement.
5. Accept the default installation directory, usually `/opt/ibm/ILOG/CPLEX_StudioXXX`.
6. Begin the installation, then wait for it to finish. Exit the installer after it's done.
7. In your `.bashrc` file in your home directory (e.g., `/home/alice/.bashrc`), add the following lines to modify the environment variables `CPLEX_ROOT` and `LD_LIBRARY_PATH`:

```
export CPLEX_ROOT="/opt/ibm/ILOG/CPLEX_StudioXXX"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${CPLEX_ROOT}/opl/bin/x86-64_linux"
```

where XXX is replaced by appropriate version numbers.

8. You may need to restart Bash in order for the modified environment variables to take effect.

On Windows, just run the bin file and accept all the default options.

2.3.3 Setting up the Python API

Starting from version 12.8.0, CPLEX has two Python APIs: the **legacy Python API**, which is a low-overhead wrapper to the core CPLEX engine, and **DOcplex**, which is a modeling library that wraps the legacy Python API together with additional constructs. While DOcplex is easier to use, it does not have direct access to some of the more advanced features of CPLEX (e.g., callbacks).⁷ See https://www.ibm.com/support/knowledgecenter/SSSA5P_latest/ilog.odms.cplex.help/CPLEX/Python/topics/cplex_python_overview.html for details.

Assuming Python is already installed (see Module 1), there are two ways to set up either (or both) of the APIs depending on whether you are using Anaconda or not:

- If you are using **Anaconda**, you can install the DOcplex API by opening a terminal and running the commands⁸

```
conda config --add channels ibmdecisionoptimization
conda install docplex
```

See https://cdn.rawgit.com/IBMDecisionOptimization/docplex-doc/master/docs/getting_started_python.html for details.

The legacy API is trickier since there is no official Conda package. The recommended method is to first install pip into the current conda environment with `conda install pip`, then use the non-Anaconda instructions to install via pip. You can verify that the package was installed by running `conda list`.

- If you are not using Anaconda, it is **highly** recommended that you install the packages with `pip/pip3`. Follow these steps:
 1. Navigate to the root CPLEX directory (usually `/opt/ibm/ILOG/CPLEX_StudioXXX` or `C:\Program Files\IBM\ILOG\CPLEX_StudioXXX\` where XXX is replaced with appropriate version numbers).
 2. From the root CPLEX directory,
 - if you are installing DOcplex, run `pip3 install python/docplex`.⁹
 - if you are installing the legacy API, run `pip3 install cplex/python/X.Y/ARCH` where X.Y is replaced by the version of Python you are using (you can find out with `python3 -V`) and ARCH is replaced by an appropriate name for the architecture. For example, if you are installing for Python 3.6 on Linux, the last argument would be `cplex/python/3.6/x86-64_linux`.

In particular, it is **not** recommended to run `setup.py` directly or modify `PYTHONPATH` as suggested by the CPLEX documentation!

⁷However, DOcplex does provide way to access the internal cplex object (from the legacy API) associated with a DOcplex model, so it is possible to use both APIs simultaneously to a certain degree.

⁸There is a `cplex` package on the channel as well, but that is for the community version of the CPLEX solver, which we obviously don't need.

⁹If you don't have Admin privileges you can run `pip3 install --user .` instead to install the package to your user site directory. You can run the command `python3 -m site` to see where your user site directory is.

You should now be able to run CPLEX's Python examples. See https://www.ibm.com/support/knowledgecenter/SSSA5P_latest/ilog.odms.cplex.help/refpythoncplex/html/overview.html for details on the legacy API and <https://ibmdecisionoptimization.github.io/docplex-doc> for details on DOcplex.¹⁰

2.3.4 Setting up the C++ API

Assuming a C++ compiler is already installed (see Module 1), there is no additional setup needed to use the CPLEX C++ API unless you are using a build system (which we will cover in Module 5). Here are two simple ways to compile CPLEX's C++ examples:

- To compile it from the Linux command line, run a command similar to

```
g++ -m64 -g -DIL_STD -o example example.cpp \  
-I${CPLEX_ROOT}/cplex/include -I${CPLEX_ROOT}/concert/include \  
-L${CPLEX_ROOT}/cplex/lib/x86-64_linux/static_pic \  
-L${CPLEX_ROOT}/concert/lib/x86-64_linux/static_pic \  
-lconcert -lilocplex -lcplex -lm -lpthread -ldl
```

(Note that \ is simply an indicator that the command is continued on the following line.)

- Alternatively, you can use a *Makefile*, which runs the compilation command for you based on some specified patterns. See the Canvas module for a minimal Makefile based on the ones provided by CPLEX and how to use it. See *Setting up CPLEX on GNU/LINUX/macOS* in **Getting Started with CPLEX** (https://www.ibm.com/support/knowledgecenter/SSSA5P_latest/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/set_up/GNU_Linux.html) for details.

See https://www.ibm.com/support/knowledgecenter/SSSA5P_latest/ilog.odms.cplex.help/refcppcplex/html/overview.html for details.

2.4 Which Should I Choose? Gurobi or CPLEX?

In terms of performance, Gurobi and CPLEX are fairly equal; while for individual problems one may be slightly faster, on average they are roughly of the same caliber. The two solvers are also fairly equal feature-wise; while at certain times one may have a feature that the other does not, such differences tend to disappear in subsequent releases. That said, there are a few distinctions as of Gurobi 8.1 and CPLEX 12.9.0:

- Gurobi's documentation is significantly more organized and easier to navigate than CPLEX's documentation.
- Gurobi has an official R API, while CPLEX only has unofficial packages (`cplexAPI` which is an R wrapper to the C API, and `Rcplex` which has a simple interface for solving models specified by matrices).

¹⁰If you have tried the recommended options and you still get import errors, as a last resort you can create a soft link to the directory containing the CPLEX package *in the same directory as the Python script that needs it*. Note that the soft link *must* be named `docplex` (for DOcplex) or `cplex` (for legacy API). You can create soft links in Linux using the command `ln -s`.

- Gurobi historically has better support for multiobjective optimization; for example, CPLEX only recently gained the ability to use hierarchical objectives for MIPs.
- CPLEX has slightly more users in both academia and industry due to its longer history.
- CPLEX, in addition to a MIP solver, comes packaged with a CP solver, a modeling language, and an IDE; Gurobi only includes a MIP solver.
- CPLEX's generic callbacks and Gurobi's callbacks have similar functionality. However, CPLEX also has legacy callbacks which provide slightly more control over the branch-and-bound process (in particular, node selection and branching rules), although they are incompatible with the default dynamic search.

2.5 Optional Content

2.5.1 Understanding CPLEX's Documentation

CPLEX's official documentation, which is part of IBM Knowledge Center, can be a little difficult to navigate if you are not used to it. Here are some tips:

- If you are looking for documentation on how to use a specific class/function, use the search bar in the top right. Otherwise, the easiest way to navigate is via the [Table of Contents](#) on the left side of the screen (if it's closed, hover over it and click on the pin icon.)
- Make sure you are looking at the documentation for the version of CPLEX you are using. This is indicated at the top of the page (e.g., [Home](#) » [ILOG CPLEX Optimization Studio 12.9.0](#) » [CPLEX](#)). You can change the version of CPLEX documentation by using the [Change version or product](#) dropdown menu in the top left section of your screen.
- If you are looking for information on a parameter setting, go to [CPLEX](#) » [Parameters of CPLEX](#), and look in [Topical list of parameters](#) or [List of CPLEX parameters](#).
- If you are looking for a high-level overview of a particular feature/concept (e.g., callbacks, Benders annotation), look under [CPLEX](#) » [User's Manual for CPLEX](#) typically in [Advanced programming techniques](#) or [Discrete optimization](#). If you want to know what function to use, [CPLEX](#) » [Overview of the APIs of CPLEX](#) can be useful.
- Most relevant documentation for the C++ API is found in [CPLEX](#) » [CPLEX C++ Reference Manual](#) » [optim.concert](#) in either the *Classes* or *Functions* section. The major exception is `IloCplex`, which is documented under `optim.cplex.cpp` instead of `optim.concert`.
- Most relevant documentation for the legacy Python API is in the `cplex._internal._subinterfaces` section. It is helpful to read the **Structure of the package cplex** section on the overview page (https://www.ibm.com/support/knowledgecenter/SSSA5P_latest/ilog.odms.cplex.help/CPLEX/Python/topics/PLUGINS_ROOT/ilog.odms.cplex.help/refpythoncplex/html/overview.html) to understand how the package is organized.
- Most relevant documentation for DOcplex is in `docplex.mp.model`, `docplex.mp.linear`, or `docplex.mp.solution` module pages. Note that DOcplex documentation is hosted on Github, NOT IBM Knowledge Center.