Chalmers University of Technology
University of Gothenburg
Mathematical Sciences
Optimization
Ann-Brith Strömberg
Edvin Åblad

MVE165
MMG631
Linear and Integer Optimization
with Applications
Exercise information
March 21, 2019

Computer exercise: Linear programming and software solvers

Introduction

The purpose of this computer exercise is to make you familiar with the use of software for computing solutions to linear programs. While performing the exercises, you will learn about the different solvers with their strengths and drawbacks. This will be helpful for the assignments and hopefully also in your future work. The following software is included:

- JuMP[1] An algebraic modelling language for linear, quadratic, and nonlinear constrained optimization problems embedded in Julia[2];
- MATLAB's optimization toolbox[3] handles linear, nonlinear unconstrained and constrained, and binary (linear) programs. It also contains special purpose solvers for quadratic programs and nonlinear least squares problems;
- Gurobi[4] a linear, integer linear, and quadratic programming solver with interfaces to MATLAB, JuMP, and Python;
- Coin-Or project[5], an open-source project containing CLP linear and quadratic programming solver, CBC a integer linear programming solver, both can be called from JuMP.

Before you do the exercise, you should read Chapters 1, 2.1, 2.6, and 4 in the course book[6].

[1] `http://www.juliaopt.org/`
[2] `https://julialang.org/`
[3] `http://www.mathworks.com/products/optimization/`
[4] `http://www.gurobi.com`
[5] `https://projects.coin-or.org`
[6] Optimization (2010)/Optimeringslära (2008), by J. Lundgren, M. Rönnqvist, P. Värbrand. Studentlitteratur

Exercise 1 – JuMP installation & testing

This exercise has two purposes:

1. Installation of necessary modelling and solving software for the course.
2. Introduce the programming language Julia and know the most basic functions of the modelling package JuMP.

Installing JuMP

Here we present how to install Julia, the add-on JuMP, and the editing environment Juno. If installing on your own system, follow the instructions on the link `http://docs.junolab.org/stable/man/installation.html`. Below, we repeat the steps from the link, with specific instructions for Chalmers' Linux system.

1. Download the programming language Julia from `https://julialang.org/downloads/`. For the Chalmers system choose 64bit "Generic Linux Binaries for x86" and extract the Julia folder into your home folder.
2. Start the editor Atom[7], which is installed on Chalmers' Linux system: type `atom` in the terminal.
3. Install the add-on Juno: in the Atom open settings (`cmd+`) and go to the tab "install". Type `uber-juno` in the search field and click install. After installation you may choose to use the standard layout.[8]

You have now successfully installed Julia and the program Atom to edit it. In the console (Julia > open console), hit "enter" and wait for it to update. Then type, for example:

```
x=[1 2 3]'
x*x'
```

Julia is based on packages, in this course we will mainly use the modelling package JuMP. Before we can use packages, we need to add them. To open the package manager, type `]` in the console. To add a package, type `add "packagename"`. Add some packages that are used in this course:

```
add JuMP#v0.18.5
add Clp
add MathProgBase
```

To close the package manager, type backspace. Now we are ready to study the introductory code[9]. In the atom menu go to "File > open" and open the folder containing your files. Begin by reading the commented main file `intro_run.jl`. To run code either select it and press `shift+enter` or `cmd+shift+enter` to run the entire file.[10] Julia uses "Just in Time compilation" (JiT), which requires some extra time the first time the code is executed. To get a more complete description, study the documentation of JuMP[11] and of Julia[12]. Note that in this course we use the legacy

[7] `https://atom.io/`

[8] You may need to add the path to Julia if it is not accessible via the command julia. To do so, go to Packages -> Julia -> Settings and in the first box for "Julia Path" insert the path to your Julia installation, e.g., `C:/Users/XXX/Julia-1.1.0/bin/julia.exe`.

[9] An implemented model of the diet problem.

[10] In the most cases, this workflow is efficient. However, if the integrated Julia console feels unresponsive, instead use a separate Julia console to execute your heavy code and still use Atom to edit it.

[11] `http://www.juliaopt.org/JuMP.jl/v0.18/`

[12] `https://docs.julialang.org/en/v1/`

version (v0.18.5) of JuMP and not the latest relase (v0.19), this due to current instability issues in the later version.

Using commercial solvers

Even though Clp is a good linear programming solver it is not the best; open source solvers are often outperformed by commercial ones. JuMP supports many solvers, and here we show how to use the solver Gurobi in JuMP. Go to `http://www.juliaopt.org/JuMP.jl/v0.18/installation` and click, for example, the link `Gurobi.jl` to get instructions on how to use Gurobi. The table is also useful for choosing an appropriate solver for your optimization problem. The model in `intro_mod.jl` is a linear optimization model; hence LP solvers apply. If, for example, some variables are declared to be integer, use instead a MILP solver.

Before we can add Gurobi to JuMP it needs to be installed, licensed, and to let JuMP know where it is located. The two first steps are already done for the Chalmers Linux system. If you are using your own computer, it is fairly straightforward; see Appendix A. To let Julia know the location of Gurobi, make sure that an "environment variable" is set:

Linux/Mac If on the Chalmers system, in the terminal, type:

```
GURPATH=$(find /chalmers/sw/sup64/ -maxdepth 1 -name "gurobi*");
echo export GUROBI_HOME=$GURPATH >> ~/.bashrc
cp $GURPATH/gurobi.lic ~/
```

If you use your own computer this should be part of the Gurobi installation; check this by

```
echo $GUROBI_HOME
```

If this is not set, do the above using your path to Gurobi, for example,

```
c:/gurobi810/linux64
```

and you don't need to copy the licence file.

Windows On your own system check that gurobi is in the path:

```
echo %PATH%
```

If it is missing, do

```
setx GUROBI_HOME "c:\gurobi810\win64
```

and restart windows.

Now you can add the Gurobi package in Julia. In the Julia console type:

```
add Gurobi
```

If all goes well you can now modify `"intro_run.jl"` to use Gurobi as solver. If the installation failed on your own computer, consort the installation guide `https://github.com/JuliaOpt/Gurobi.jl#installation`.

Exercise 2 (optional) – MATLAB

Solve the simple linear programming problem

$$\begin{aligned} \text{(LP1)} \qquad \text{minimize} \quad & z = -x_1 - 2x_2, \\ \text{subject to} \quad & -2x_1 + x_2 \leq 2, \\ & -x_1 + x_2 \leq 3, \\ & x_1 \leq 3, \\ & x_1, \; x_2 \geq 0, \end{aligned}$$

graphically. Then, start MATLAB by typing '`matlab &`' in a Linux command window. Implement and solve (LP1) using `linprog` in MATLAB. The problem does not need to be in standard form, however it must be in matrix form. Try `'help linprog'`; it can handle equality constraints (`Aeq` and `beq`), inequality constraints (`A` and `b`), and lower (`lb`) and upper bounds (`ub`) on variables. Bounds on variables can be modelled using general linear inequalities, but it is often more efficient to model these explicitly. This is extra important for large problems.

With the structure `options`, the user can influence the algorithm. Try `'help optimoptions'` to see a list of options. To set any of them, e.g., `Display` and `MaxIter`, generate the structure '`options`' by

```
> options = optimset('Display', 'iter', 'MaxIter', 100);
```

and solve using (see also `'help linprog'`)

```
> [x,f] = linprog(c, A, b, [], [], lb, [], x0, options)
```

where `x0` refers to a starting point (which may be omitted, using `[]`) and `options` is either set or omitted.

For solving other problem types, as e.g., quadratic programs, try `'help optim'` to see a list of the solver routines in the toolbox.

A Install Gurobi on your own computer

1. Create an account at `http://www.gurobi.com/registration/general-reg`.
2. Download Gurobi at `http://www.gurobi.com/downloads/gurobi-optimizer`
3. Request a licence at `https://user.gurobi.com/download/licenses/free-academic`.
4. Goto `https://user.gurobi.com/download/licenses/current`
   (a) select the license
   (b) copy the terminal command `grbgetkey` .....
5. Extract/install the Gurobi-file from step 2
6. Execute the terminal command: `grbgetkey` .....
   (You need internet connection via Chalmers Domain, e.g., at campus or by vpn.)