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

MVE165
MMG630
Applied Optimization
Exercise information
March 17, 2010

Exercise 1: 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 softwares are included:

- MATLAB's optimization toolbox[1] handles linear, nonlinear unconstrained and constrained, and binary (linear) programs. It also contains special purpose solvers for quadratic programs and nonlinear least squares problems.
- AMPL[2], an algebraic modelling language for mathematical programming;
- CPLEX[3], a linear, integer linear, and quadratic programming solver with interfaces to AMPL and MATLAB;
- GLPK[4] (GNU Linear Programming Kit), a linear and integer linear programming solver with an interface[5] to MATLAB;
- CLP[6], a linear and quadratic programming solver with an interface[7] to MATLAB.

Before you do the exercise you shoud read Chapters 1, 2.1, 2.6, and 4 in the course book[8].

[1] `http://www.mathworks.com/products/optimization/`
[2] `http://www.ampl.com/`
[3] `http://www-01.ibm.com/software/integration/optimization/cplex/?S_CMP`
[4] `http://www.gnu.org/software/glpk/`
[5] `http://glpkmex.sourceforge.net/`
[6] `https://projects.coin-or.org/Clp`
[7] `http://control.ee.ethz.ch/~joloef/clp.php`
[8] Optimization (2010)/Optimeringslära (2008), by J. Lundgren, M. Rönnqvist, and P. Värbrand. Studentlitteratur.

Exercise 1.1 – MATLAB

Solve the simple linear programming problem

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

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 modlled 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', 'on', '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; to choose between different linear programming solvers (simplex and interior-point), type e.g.:

```
>> options = optimset('Simplex', 'on', 'LargeScale', 'off');
```

For this simple example, no difference between the methods is seen, but for larger problems the difference is huge. The simplex method implemented in `linprog` seems to be more robust than the interior-point method (see Ch. 7.5 of the book by Lundgren et al.). The simplex implementation in `linprog` cannot, however, handle sparse matrices efficiently, which makes it very slow for large problem instances.

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

Exercise 1.2 – AMPL

A nonferrous metals corporation manufactures four different alloys from four basic metals. The objective is to determine the optimal product mix to maximize gross revenue without exceeding the supply limits. The requirements are given in the table below. Formulate a linear optimization model for this problem, implement the model in AMPL and solve it using CPLEX.

Metal	Proportions of metal in alloy				Total supply of
	1	2	3	4	metal/day
1	0.25	0.6	0.2	0.1	5 tons
2	0.25	0.2	0.6	0.7	5 tons
3	0.25	0	0.1	0.1	1 ton
4	0.25	0.2	0.1	0.1	2 tons
Selling price of alloy/ton:	\$30	\$15	\$25	\$23	

`AMPL` works with three files; a script file (`name.run`), a model file (`name.mod`), and a data file (`name.dat`). The variable names, the parameter names, the objective function, and the constraints are formulated in the model file. All the data (e.g., the entries of the cost vector c) are specified in the data file. The solution course is indicated in the script file.

The files `lp1.run`, `lp1.mod`, and `lp1.dat` (for solving the model (LP1)) can be downloaded from the course home page `http://www.math.chalmers.se/Math/Grundutb/CTH/mve165/0910/`. To solve the above model, modify these files and solve the problem by typing[9] at the prompt (presuming that your script file is named `alloy.run`):

```
> ampl alloy.run
```

Transforming the linear program into an integer linear program is made by adding '`integer`' to the variable declaration in `lp1.mod`, according to '`var xJ integer >= 0;`'.

[9] If you want to use your own computer, download a student version of `AMPL`.

Exercise 1.3 – NETLIB

In this exercise the MATLAB optimization toolbox, GLPK, CLP, and CPLEX will be used to solve test problems from NETLIB[10] repository, which contains a large collection of numerical software. The repository also contains a linear programming test set, which has been used extensively for benchmarking linear programming solvers. The problems are small scale according to the standard of today, but many of them are nonetheless quite difficult to solve. The test problems have been submitted by several researchers; many of them are based on real applications. Other software solvers exist, e.g., Gurobi[11] and SCIP[12].

From the course web page, download the recommended NETLIB examples `lpcoll.zip` (or the whole NETLIB repository `netlibfiles.zip`) and unzip it using the command '`unzip lpcoll.zip`' (or '`unzip netlibfiles.zip`'); all these examples are stored in MATLAB's `.mat`-format.[13] To load a test example into MATLAB, type e.g., '`load agg3.mat`' in a MATLAB command window. Also, download the text file `lp_data_readme` containing information on the sizes and optimal values for the problems. Each `.mat`-file contains the matrices `A`, `b`, `c`, `lo`, and `hi`, and `z0` for a linear program on the form

$$\min z := c^{\mathrm{T}}x + z_0, \qquad \text{subject to} \quad Ax = b,\ \mathrm{lo} \le x \le \mathrm{hi}.$$

Compare the performance of the solvers on the NETLIB linear programming test set. Measure the computation time and whether the correct optimum is found. You don't have to try all problems; pick a few of varying size. Try to draw conclusions about the efficiency of the solvers applied to the different test problems.

- To run GLPK, type (in MATLAB):

  ```
  >> addpath /chalmers/sw/unsup/glpkmex-4.38/mex
  ```

  The driver routine is called `glpk`. Try '`help glpk`'. For problems with only equality constraints, as for the NETLIB problems, the vector `ctype` should be all '`S`', according to the MATLAB commands (assign appropriate values to the parameter `vartype` analogously):

  ```
  >> for j=1:size(A,1) ctype(j) = 'S'; end
  ```

- To run CLP, type (in MATLAB):

  ```
  >> addpath /chalmers/sw/unsup/clp-1.6.0
  ```

  The driver routine is called `clp`. Try '`help clp`'.

- To run CPLEX, type (in MATLAB):

  ```
  >> addpath /chalmers/sw/unsup/cplexmex/dist
  >> setenv('ILOG_LICENSE_FILE', ['/chalmers/sw/sup/cplex-12.1/ilm/access.ilm'])
  ```

  The driver routine is called `cplexmex`. Try '`help cplexmex`' and '`help cplexmexparams`'.

[10] `http://www.netlib.org/`
[11] `http://www.gurobi.com/`
[12] `http://scip.zib.de/scip.shtml`
[13] To download the examples on other formats, see `http://www.math.ufl.edu/~hager/coap/format.html`