Advanced Automation and Control

Laboratory 2

Prof. Davide M. Raimondo with the collaboration of Dr. Marcello Torchio

Mixed Integer Linear Programming

Problem description

The production process of smart washing machines at the ACME company can be conducted in three different ways: (*i*) manually, (*ii*) semi-automatically, and (*iii*) automatically. *However, the acquisition of new technology such as the semi-automatic or the automatic one requires a fixed cost of respectively 2000 (semi-automatic) and 4500 (automatic) euros.* Each of the production approaches involves the allocation of different amounts of human resources. In particular, the manual production demands 2 minute of qualified work, 40 minutes of non-qualified work and 4 minutes of assemblage. If the semi-automatic solution would be chosen, 4 minutes of qualified work, 20 minutes of non-qualified work, and 8 minutes for the assemblage would be required. Finally, 8, 10 and 12 minutes respectively would be required for the automatic method. ACME has a pool of 4000 minutes of qualified work, 8000 minutes of non-qualified work and 12000 minutes of assembly. The production costs of a washing machine are 45 euros if produced manually, 40 euros if produced semi-automatically, and 35 euros if produced automatically. Each smart washing machine is sold at 90 euros. From a commercial point of view, the ACME company is interested in the following objective:

Find the optimal number of washing machines to be produced in order to maximize the profit

A mixed integer linear program can be derived in order to support the decision-making process at ACME.

Problem formulation

The optimization variables of the problem are x_1, x_2 and x_3 (continuous) which are used to represent the number of washing machines produced using the manual, semi-automatic, and automatic methods respectively and δ_2, δ_3 (binary) which are used to indicate whether a certain technology is acquired or not. In particular

$$\delta_2 = 1 \leftrightarrow x_2 > 0$$

$$\delta_3 = 1 \leftrightarrow x_3 > 0$$

The optimization problem can be formulated as follows

$$\max_{x_1,x_2,x_3,\delta_2,\delta_3} 85x_1 + 65x_2 + 55x_3 - 2000\delta_2 - 4500\delta_3$$

subject to

$$\begin{aligned} 2x_1 + 4x_2 + 8x_3 &\leq 4000 \\ 40x_1 + 20x_2 + 10x_3 &\leq 8000 \\ 4x_1 + 8x_2 + 12x_3 &\leq 12000 \\ \epsilon\delta_2 - x_2 &\leq 0 \\ \epsilon\delta_3 - x_3 &\leq 0 \\ x_2 - U_2\delta_2 &\leq 0 \\ x_3 - U_3\delta_3 &\leq 0 \\ x_1, x_2, x_3 &\geq 0 \\ \delta_2, \delta_3 &\in \{0, 1\} \end{aligned} \quad .$$

where U_2 and U_3 are upper bounds on the optimization variables x_2 and x_3 (please compute the tightest upper bounds for such variables).

The solution of the problems above has been addressed in the scripts:

- **MILP_example_glpk.m** (it relies on the MATLAB function *glpk* for solving MILPs)
- **MILP_example_yalmip.m** (it makes use of *Yalmip*, a free MATLAB Toolbox for rapid prototyping of optimization problems)

Both scripts make use of the *MPT Toolbox* which, among the many features, includes Yalmip and allows the plotting of polyhedra. To execute the scripts use the commands:

- `[x,fval,exitflag]=MILP_example_glpk(1)`
- `[x,fval,exitflag]=MILP_example_yalmip(1)`

Both functions accept one argument which represents a flag that if set to *1* allows the plotting of isocost lines. The ouput arguments x_{out}, δ_{out}, $fval$ and $exitflag$ provide respectively the optimal value of the continuous and binary variables, the optimal cost and a flag indicating whether the optimization was completed successfully or any problem occurred (problem not feasible, unbounded, etc.). Note that the meaning of *exitflag* is different when provided by yalmip or glpk (see the scripts for further details).

In order to account for the presence of binary variables, Yalmip relies on the function *binvar*. *Glpk* has a structure similar to linprog but accepts among its arguments the parameter *vartype*. This latter is a string which contains a 'C' or a 'B' to indicate that a certain variable is continous or discrete. In our case, for example, the string 'CCCBB' would indicate that x_1, x_2, x_3 are continuous while δ_2, δ_3 are binary. Please use *help binvar* and *help glpk* for further information.

Both scripts make use of the function Polyhedron (use *help Polyhedron* to see how it works) in order to plot feasible sets and iso-cost lines. The scripts come with detailed comments (use the editor to see inside the files) which should help the reader in understanding the routines. The scripts return also the time required by glpk and yalmip to solve the MILP programs. While Yalmip is quicker to program, it is usually slower in solving the optimization (conversion overhead).

Using the *glpk* script and then the *yalmip* one:

1. Solve the optimization problem. Are the obtained outcomes meaningful? Why? Could you explain why the feasible sets are so different depending on the values of the binary variables?
2. Solve now the problem with 3500 minutes as maximum number of qualified work. What happens now? Is the result meaningful? Please replace *sdpvar* with *intvar* in Yalmip and 'C' with 'I' in the *vartype* of *glpk*. These modifications allow to specify that x_1, x_2, x_3 are integer. Are the obtained results now meaningful?

3. Assume now that if the number of produced units goes over 400, then the plant needs an expansion of the production site which comes at a price of 5000 euros. Please reformulate in Yalmip the optimization problem taking into account this extra problem. What is now the optimal solution?

4. Assume now the price for the site expansion is of 8000 euros. What happens in this case?

5. Please formulate the problem with the expansion site costs in *glpk* (this implies the reformulation of the logical statement into mixed-integer constraints.