ECE 695 Homework Report

School:	Purdue
Course Title:	ECE 69500
Instructor:	Prof Aliprantis
Homework:	4
Name:	Yun Zhi Chew

Contents

0 Nomenclature

MILF	Mixed Integer Linear Formulation
MILP	Mixed Integer Linear Programming
BPS	Bulk Power System
RHS	Right Hand Side

1 Executive Summary

The overall objective of this homework is to implement a MILF method for a Thermal Unit Commitment Problem that is described in the IEEE paper entitled:

"A Computationally Efficient Mixed-Integer Linear Formulation for the Thermal Unit Commitment Problem"
Miguel Carrión, *Student Member, IEEE*, and José M. Arroyo, *Senior Member, IEEE*

The specific objectives are as follow:

- Study and understand the constraint equations described in the paper
- Replicate the MILF equations in MILP format
- Run the simulation using a predefined MILP Function from MATLAB or similar software
- Compare simulation results with those presented in the paper
- Attempt different algorithmic options to optimize code, simulation time and/or results
- Discuss learning points

2 Background

The paper in question presents a new MILF for the Unit Commitment problem of Thermal Units.

Unit Commitment is a classical problem for any BPS. Unlike the Economic Dispatch problem, we do not assume that all the units are already connected to the system. Briefly, we can regard Economic Dispatch as a sub problem within the Unit Commitment problem.

A Thermal Unit can be regarded as a single turbine in a power plant. Typically, depending on the capacity of the plant, a power plant would contain more than one Thermal Unit.

Note: User should not confuse the term "Thermal Unit" with the term "British Thermal Unit (Btu)'.

3 Algorithm

Using the new MILF, the Unit Commitment problem of Thermal Units is numerically solved via the classical MILP algorithm.

The MILP algorithm solves the problem:

```
MINIMIZE      f' * x
SUBJECT TO    A * x     <= b
              Aeq * x == beq
              lb        <= x <= ub
```

LEGEND

f':	transpose of a Vector	=> Coefficients of the objective function
x:	Vector	=> Variables

A:	Matrix	=> Coefficients of the inequality constraints
Aeq:	Matrix	=> Coefficients of the equality constraints
b:	Vector	=> RHS of Inequality
beq:	Vector	=> RHS of equality
lb:	Vector	=> Lower bound conditions
ub:	Vector	=> Upper bound conditions

4 Numerical Tools

As the number of equations (equivalent to the number of rows in Matrix A and Aeq) are large, it is probably unfeasible to solve this problem by hand.

For the purpose of this homework, we shall employ the use of the following:

Computing Software: MATLAB
Computing Toolbox: Gurobi

The constraint equations described in the paper are formulated in MATLAB. This will be elaborated in the following sections.

We opt to call the LP solver from GUROBI using the MATLAB GUROBI interface:

```
function [x, fval, exitflag] = gurobi_intlinprog(f, intcon, A, b, Aeq, beq, lb, ub)
```

The source code for the function above can be downloaded from GUROBI's website.

5 Methodology

The solution to this or any MILP problem can be subdivided into two sub-stages. This shall be elaborated in the next two sections.

6 Initialization

The first sub-stage is Initialization. In this stage, we declare and define all the variables, constants and parameters that are required to solve the MILP problem. They shall be collectively known as elements.

These elements are declared inside an array for ease of reference.

Since most elements are not dependent on time, the elements are referenced according to the index of the thermal unit, denoted as j.

Example

UT_j is defined as the minimum up time of unit j

In MATLAB, UT_j is declared as UT(j), where UT(1) refers to the minimum up time of unit 1

Variables

The variables are the unknown values that we are trying to compute, given the minimization or maximization problem.

All the variables are listed on the first page of the paper (pg. 1371).

Constants

The constants are the coefficients of the various constraint equations that reduce the feasible region of the MILP solution.

These constants are listed on the first page of the paper (pg. 1371).

Parameters

Lastly, the parameters refer to the material operating conditions of the power system and the individual thermal units.

These parameters are given in Table A, B and C on page 1377.

7 Formulation

Once all the elements have been initialized, the constraint equation sets can be formulated.

Number of Equation Sets

The paper formulated 26 equation sets in all:

• Objective Equation	1
• “Subject to “ Equations	2, 3, 4
• Production Cost	5, 6, 7, 8, 9, 10, 11
• Startup Cost	12, 13
• Shutdown Cost	14, 15
• Generation Limits	16, 17
• Ramping Limits	18, 19, 20
• Minimum Up Time Constraint	21, 22, 23
• Minimum Down Time Constraint	24, 25, 26

For the purpose of this homework and based on the assumptions given in the paper, the following equation sets are omitted:

4, 5, 14,15,18,19,20,21 and 24

Algorithm

Each equation sets expand out to varying number of equations, depending on the number of units and time that are relevant.

It is theoretically possible to type and list out all the many thousands of equations, However, it would be tedious, time consuming, ineffective and prone to human errors.

Instead, each set of equations are expanded out by means of for loops. Since equation set 2 expands out to only 24 equations, we will illustrate this method using the MATLAB code for equation set 2

```
equation        = 2;
starting_index  = starting_index_3;           % Variable Index
A_counter       = 0;                          % Variable counter for eqn set
A_row_counter   = 0;                          % Row counter for eqn set

for k = start_time:end_time

    A_row_counter = A_row_counter + 1;                % Increment counter
    equation_B_counter = equation_B_counter + 1;      % Does not reset

    % Right Hand Side
    B(equation_B_counter) = - D(k);                   % Convert >= to <=

    % Left Hand Side
    for j = 1: total_no_of_units
        A_counter = A_counter + 1;                    % Increment counter
        A_entry_counter = A_entry_counter + 1;        % Does not reset

       % Each row represents a particular time
       row(A_entry_counter)  = equation_B_counter;
       % Each column represents a unit in time
       col(A_entry_counter)  = starting_index - 1 + (k-1)*total_no_of_units + j;
       val(A_entry_counter)  = - 1;                   % Convert >= to <=

end % j loop

    % After assignement, print status
    if k == end_time,
        ending_index = starting_index - 1 + A_counter;
        printequation(equation, A_counter, A_row_counter, ending_index,
equation_B_counter);
    end

end % k loop
```

Debugging

Due to the large number of equations that will be generated, we need to a way to identify potential errors or bugs.

"printequation" is a user-defined function to output the number of variables and rows on the command window. It is used to verify that the correct number of equations or rows are generated.

8 Documentation

As illustrated by the small extract of the MATLAB code in the previous section, proper documentation and comments are essential to ensure that the code remains readable.

Comments can be added as a **block** before a series of code or **inline** to explain a particular line of code.

Block Comments

Block Comments are typically added before the start of a code section that span multiple lines. The purpose of block comments is to provide sufficient information about what the code section is going to accomplish.

For example, the following block comment is appended before the start of every code section that expands each equation set:

```
% ---------------------------------------------------------------------- %
% Formulation of Equations Set (2)
% ---------------------------------------------------------------------- %
% Type:          INEQUALITY - CHANGE SIGN
% Description:   Summation of Power Output >= Load Demand
% ---------------------------------------------------------------------- %
% Form:          Summation [p(j)(k)] >= D(k)
%                1 * p(1)(1) + ... + 1 * p(10)(1) >= D(1)        eqn 1
%                1 * p(1)(2) + ... + 1 * p(10)(2) >= D(2)        eqn 2
%                ...
%                1 * p(1)(24)+ ... + 1 * p(10)(24)= D(24)        eqn 24
% # of eqns:     k (= 24)
```

This block comment provides useful information about the code below. Because the equations are also expanded out, it serves as a useful debugging tool.

Inline Comments

Inline comments are added at the end of each line of code to provide reminders about that specific line of code.

9 Program Code

The solution to the MILF problem is coded In 3 matlab files.

1. ECE695_HW4_MILF
2. Printequation
3. Printequation2

The file "ECE695_HW4_MILF" should be run in the same folder that contains "Printequation" and "Printequation2".

"Printequation" and "Printequation2" are simply subroutines for debugging and status purposes.

10 Learning Points

This homework taught me to appreciate the importance of algorithm formulation of mathematical equations. It is a skill that I believe would be useful in both academia and industry.

As most simple engineering problems can be described by linear equations, the proper formulation of these equations will enable us to solve and optimize many engineering problems that we may encounter in the future.

Moving forward, it would be interesting to learn how to better model and represent engineering problems as linear equations.

11 References

[1] IEEE Transactions on Power Systems, Vol. 21, No. 3, August 2006
[2] GUROBI Documentation