COMP3217 Option A - Detection of Manipulated Pricing in Smart Energy CPS Scheduling Report

The Problem

The problem considers a community of 5 users, each of whom own 10 smart home appliances. Users will have varying energy consumption over a 24 hour period and their energy usage will often peak at the same time as other users. Energy providers set higher prices during those peak energy hours to discourage usage and therefore balance the energy load. Therefore a smart home scheduler is required to compute a scheduling solution which can reduce the cost to the user. This energy scheduling solution can be computed through linear programming and determines when each smart appliance should be launched and for how long the task should be carried out.

However, some of the pricing curve guidelines for energy costs over a 24 hour period are abnormal. It is important to detect these so that the smart home scheduler can adjust the scheduling solution accordingly.

Predicting the Labels

In order to predict the labels for the testing data given, I implemented the Gaussian Naïve Bayes (GNB) classifier using Python. To decide which classifier to use I created a test script and imported various classifiers from 'scikit learn' to determine which would produce the best results. I tried the following classifiers using 80% of the training data to train the model and validating it using the other 20% of the training data:

- Multilayer perceptron
- Random forest
- K nearest neighbour
- Gaussian naïve bayes
- Support vector

Each classifier is trained using 80% of training data and then validated using the remaining 20%. The accuracies for each classifier were calculated using the 'classification_report' from 'scikit learn' metrics and are displayed in the table below:

Classifier	*Accuracy*
Multilayer perceptron	0.80
Random forest	0.91
K nearest neighbour	0.80
Gaussian naïve bayes	0.95
Support Vector	0.95

The code for this is in the file named 'sklearn_algorithm_comparison.py'. The most accurate classifiers were the Gaussian naïve bayes and the support vector. I chose to implement the Gaussian naïve bayes classifier since it was faster to run.

A GNB classifier is based on Bayes' Theorem and uses a probabilistic approach [1]. It assumes that the features are strongly independent from each other [2]. The GNB algorithm assumes that the continuous features of within each class are distributed according to a Gaussian distribution and therefore, the probability of a test feature (x_i) given a class (y) is:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)}$$

Where μ represents the mean and σ represents the variance of the whole dataset. The conditional probability of each class (in this case, the two classes representing the labels 0 or 1) is calculated using Bayes Theorem:

$$P(y_i \mid x) = \frac{P(x \mid y_i)\, P\,(y_i)}{\sum_j P(x \mid y_j)\, P(y_j)}$$

The above equation is calculated for all classes and the class with the highest probability is predicted as the label [1].

My own implementation of the GNB algorithm is stored in the file named 'training_guideline_model.py'. This file contains all the functions used to implement the algorithm. The model was trained with the whole training dataset and tested on the same dataset to calculate the training error. To calculate the validation error, the training data was split in to 80% to train the model and 20% to validate. To predict the labels for the testing data, the model was trained using the full training dataset to ensure that the labels were as accurate as possible. The testing data is read and then written to a new file named 'TestingResults.txt', where the label is appended to each line representing a guideline pricing curve. The following accuracies were achieved:

Training Accuracy:

	precision	recall	f1-score	support
0	0.94	0.95	0.94	5000
1	0.95	0.94	0.94	5000
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

Validation Accuracy:

	precision	recall	f1-score	support
0	0.94	0.96	0.95	964
1	0.96	0.94	0.95	1036
accuracy			0.95	2000
macro avg	0.95	0.95	0.95	2000
weighted avg	0.95	0.95	0.95	2000

Since both the testing and training accuracies were high, I would expect that most of the abnormal guidelines have been identified.

The labels for each guideline pricing curve are as shown below. The number in the guideline column refers to the row number of the guideline in the 'TestingResults.txt' file.

Guideline	*Label*
1	0
2	0
3	0
4	0
5	1
6	1
7	0
8	0
9	1
10	0
11	0
12	1
13	1
14	0
15	1
16	1
17	1
18	1
19	1
20	1
21	0
22	1
23	0
24	0
25	0
26	1
27	0
28	1
29	0
30	1
31	1
32	1
33	1
34	0
35	1
36	0
37	0
38	1
39	1
40	0
41	1
42	0
43	0
44	1
45	1
46	1
47	0
48	1
49	1
50	0
51	1
52	0
53	0
54	1
55	1
56	1
57	0
58	1
59	1
60	0
61	0
62	1
63	1
64	1
65	1
66	0
67	0
68	1
69	1
70	1
71	0
72	0
73	0
74	1
75	1
76	0
77	1
78	0
79	1
80	1
81	0
82	1
83	0
84	1
85	1
86	1
87	1
88	1
89	1
90	1
91	0
92	0
93	0
94	1
95	1
96	1
97	0
98	0
99	0
100	0

The Gaussian Naïve Bayes classifier predicted that 56 of the 100 predictive guideline pricing curves in the testing data provided were abnormal.

As shown above the training accuracy table shows that the F1-score for guidelines labelled as abnormal is 0.94. Therefore, I would expect that of the 56 guidelines identified as abnormal, 52.64 of those guidelines have been identified correctly.

Linear Programming Based Energy Scheduling Solution,

A python script named 'objective_function_generator.py' was created to generate the linear programming (.lp) files for each of the abnormal guidelines identified by the GNB algorithm, these are stored in a subdirectory named 'lp'. Since multiple users are considered independent from each other, a linear programming file was generated for each user for each guideline.

The linear programming file generated for each user for each guideline consists of the objective function and a number of constraints. Each user has 10 tasks that need to be carried out within the 24 hour period. Each task has a 'Ready Time' and a 'Deadline' and therefore has to be carried out within this time period; the task also has an associated 'Maximum scheduled energy per hour' and 'Energy Demand'. For example a task is defined as:

User & Task ID	*Ready Time*	*Deadline*	*Maximum scheduled energy per hour*	*Energy Demand*
user1_task1	20	23	1	1

Since this task has to be carried out between the hours of 20 and 23, has a maximum scheduled energy per hour of 1 and an energy demand of 1, the constraints for this variable 'x1' would be defined as:

0<=x1_20<=1;
0<=x1_21<=1;
0<=x1_22<=1;
0<=x1_23<=1;
x1_20+x1_21+x1_22+x1_23=1;

Where x1_XX represents task 1 being run within that hour. The objective function is formulated by multiplying each of the x1_XX variables by the unit cost for that hour according to the chosen guideline, for example:

Objective function: X.XXXXX * x1_20+ X.XXXXX * x1_21+ X.XXXXX * x1_22+ X.XXXXX * x1_23…

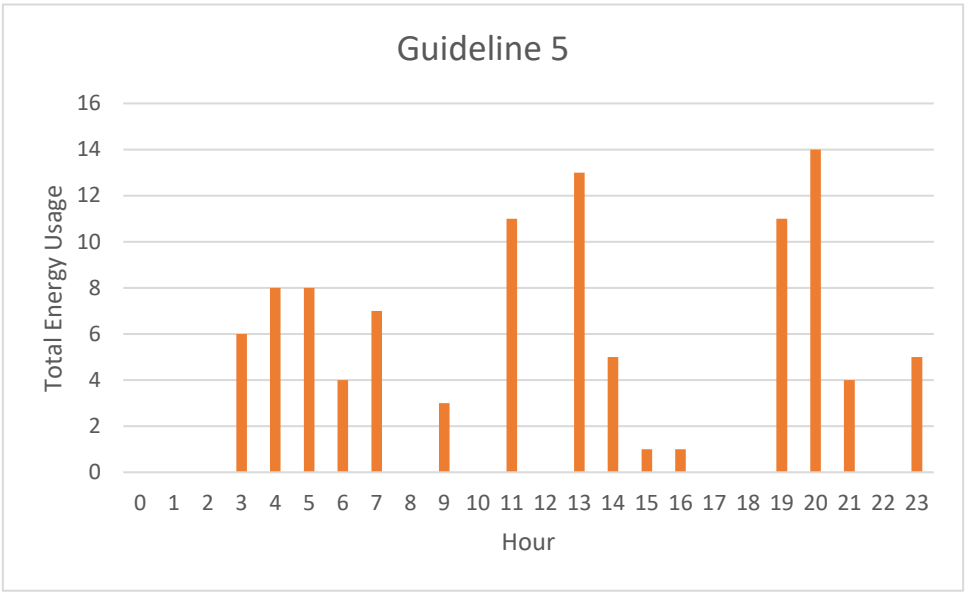
Where X.XXXXX represents the unit cost of energy during that hour. This is repeated for every task the user has to complete and this completes the linear programming file. Since the scheduling solution aims to reduce the cost to the user, the objective function is minimised.

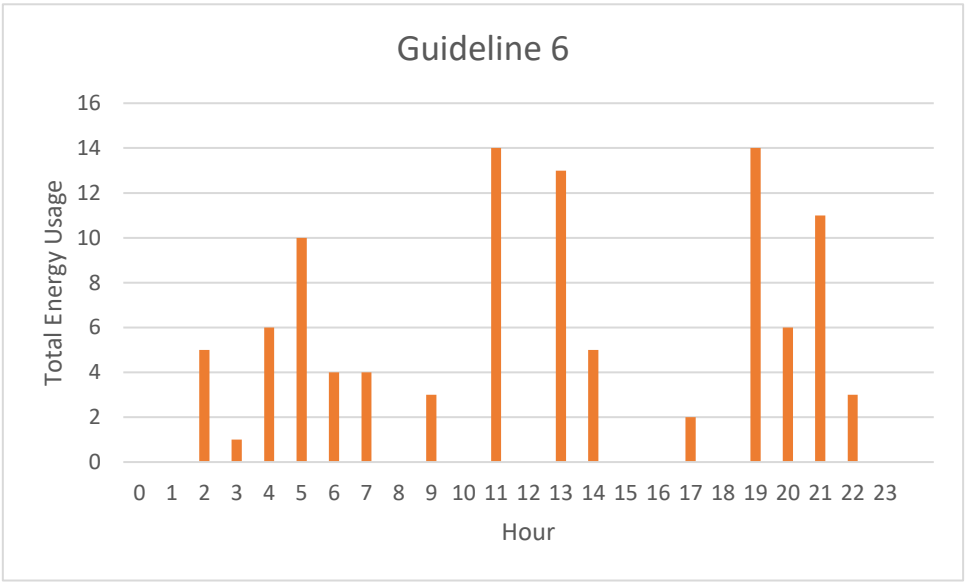
The LP_Solve application uses the revised Simplex algorithm to solve the given linear programming problem. The revised Simplex algorithm is mathematically equivalent to the Simplex algorithm but it represents the constraints as a basis of a matrix. To start, the Simplex algorithm finds a basic feasible solution that satisfies all the constraints of the problem. The algorithm moves the solution along the edges of the polyhedron representing the expanded linear constraints in R^n hyperspace, towards a better solution for the objective function [3].

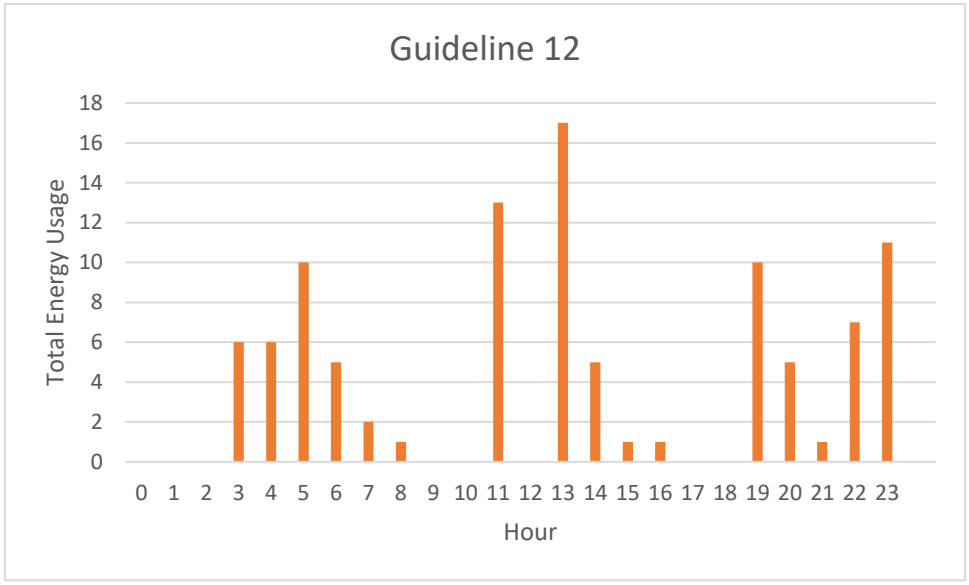
A batch file, 'solveAllLP.bat', was run to solve each of the linear programs using the command line interface for the 'LP_Solve' application and save the results to a corresponding .csv file in a subdirectory named 'lpcsv'. The batch file then placed all the different files for each user in a folder that corresponded to the guideline. This was done so that it was easy to combine all the user csv files into one .csv file for the guideline and place this in the parent directory.

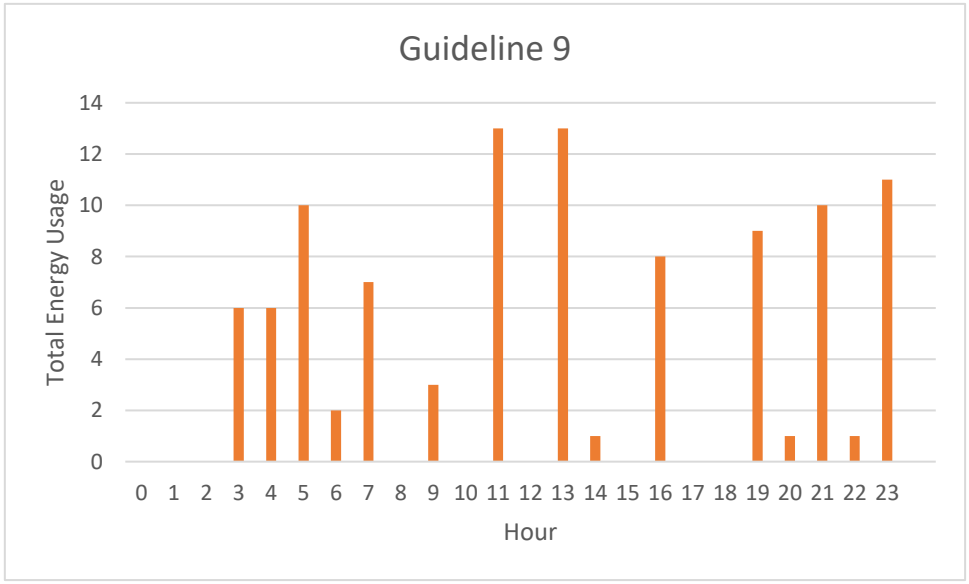
To make it easier to produce graphs, I wrote a Visual Basic script, 'csv2xlsm.vbs', to convert a .csv file to a .xlsm file and used the batch script to run this on all .csv files in the current directory. Having all of the guideline data files in .xlsm format meant that I was able to use Visual Basic Macros to clean the data and produce a graph and therefore run the macros on each of the guidelines quickly and easily.

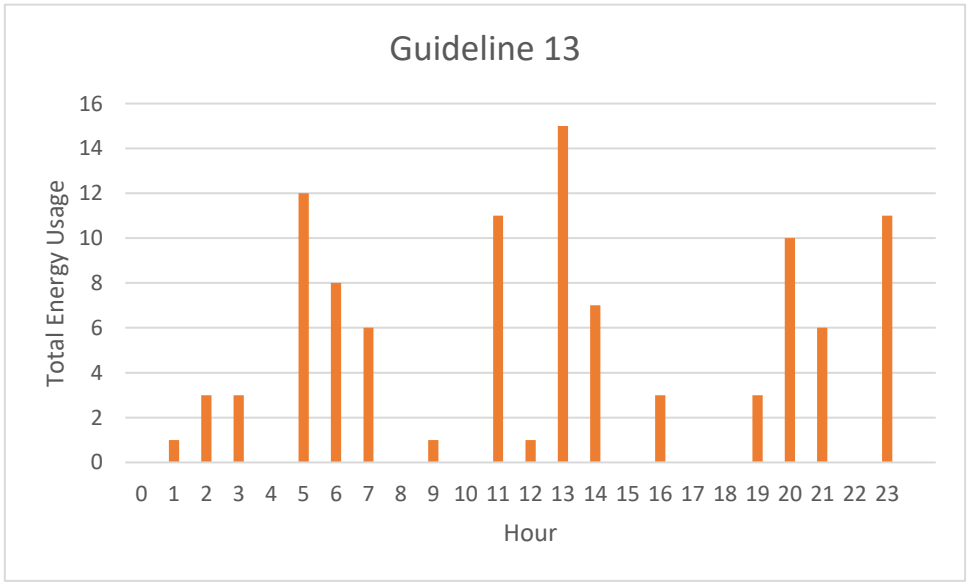
The graphs for each of the guidelines labelled abnormal (1) are shown below. The guideline number refers to the row number of the guideline in the 'TestingResults.txt' file.

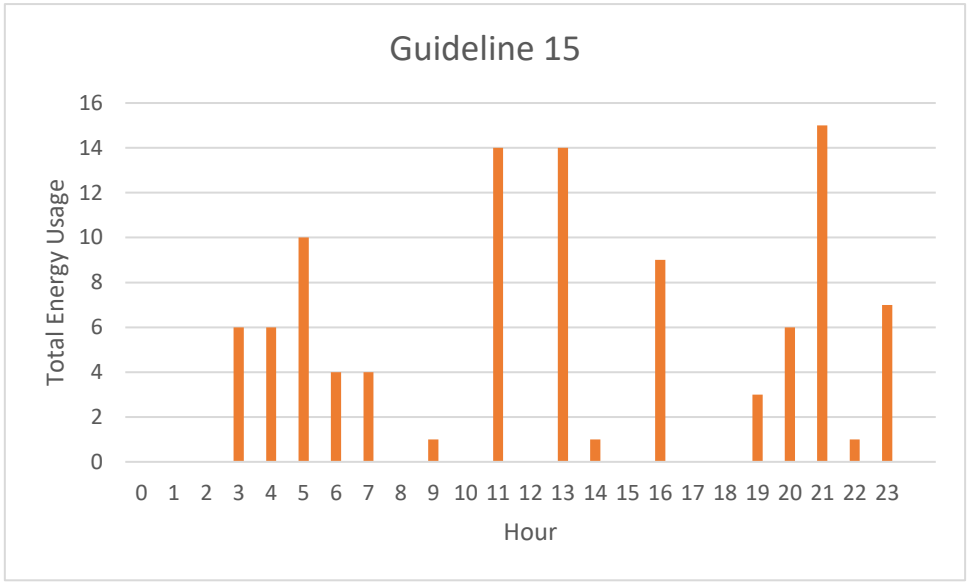


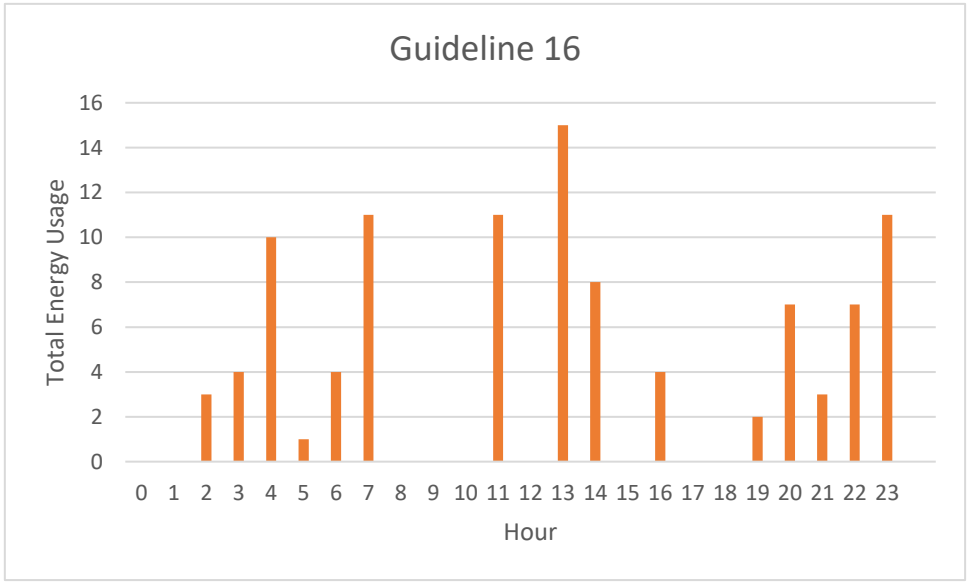


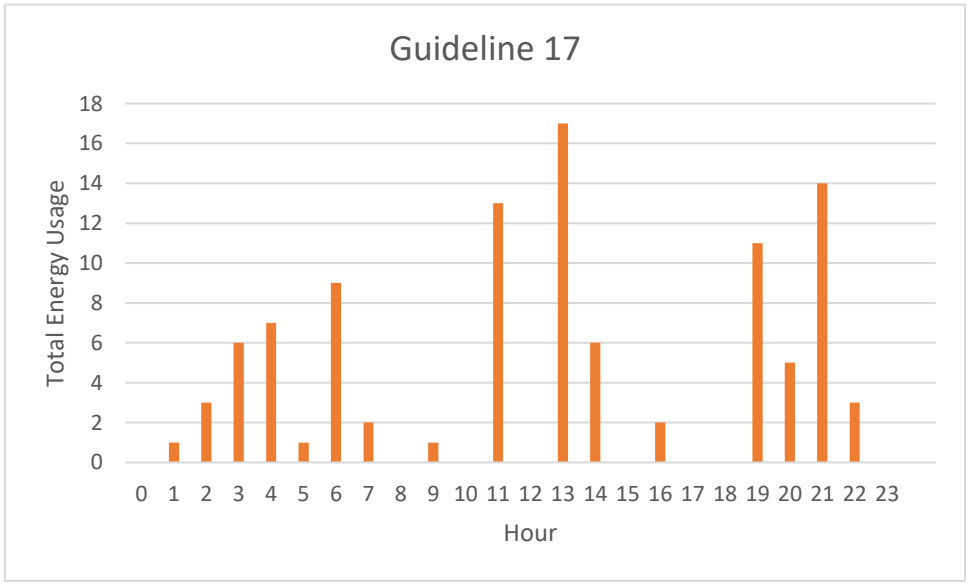

Guideline 12
Total Energy Usage
18
16
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

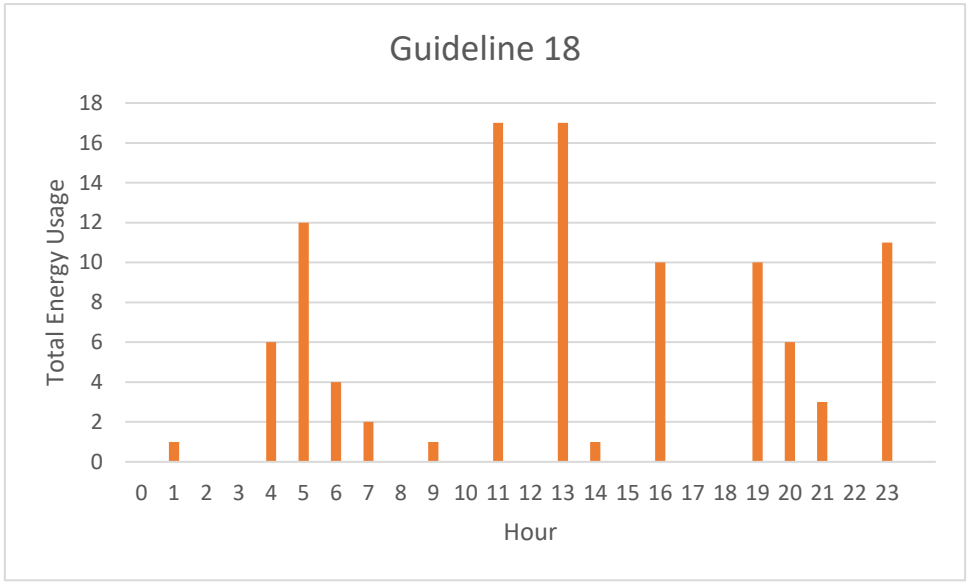

Guideline 9
Total Energy Usage
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

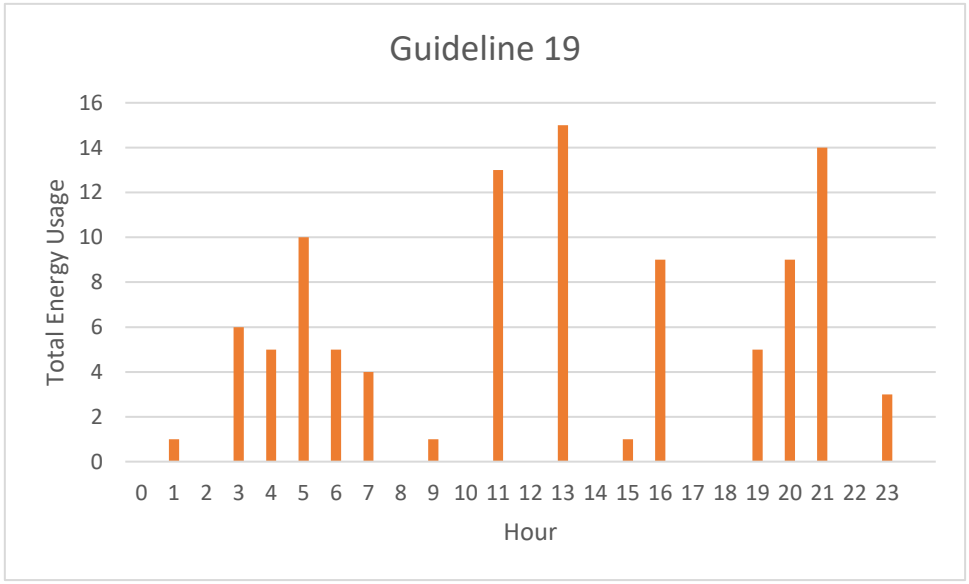

Guideline 13
Total Energy Usage
16
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

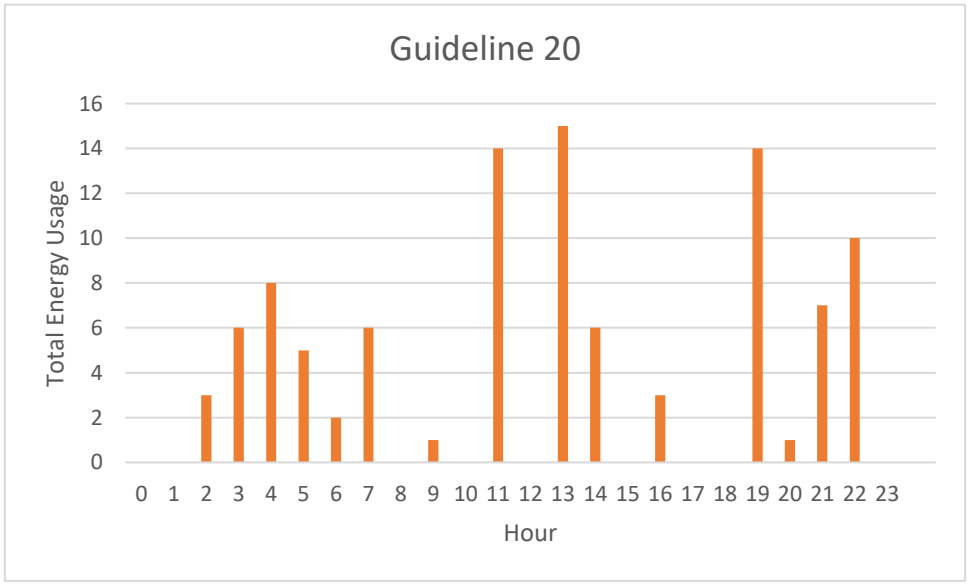

Guideline 15
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

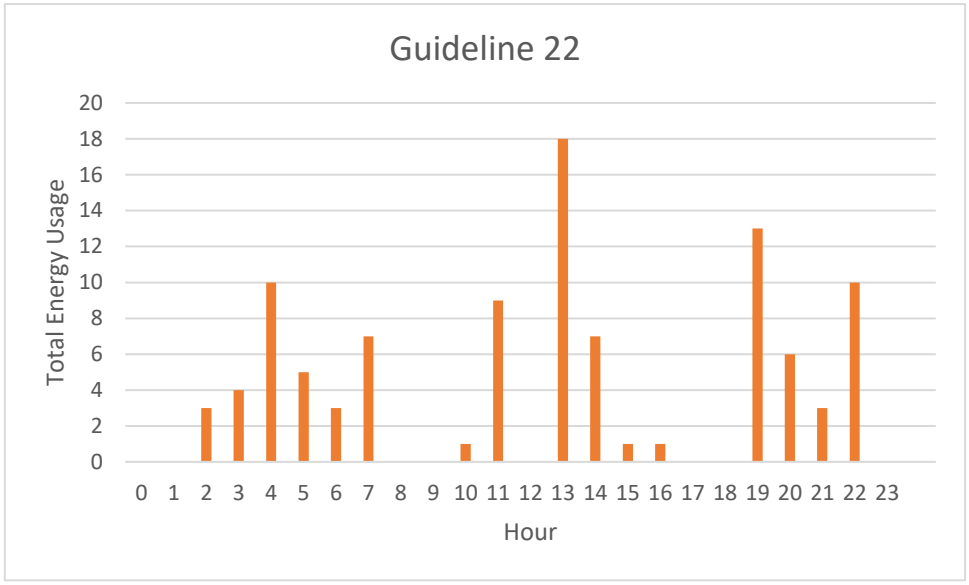

Guideline 16
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

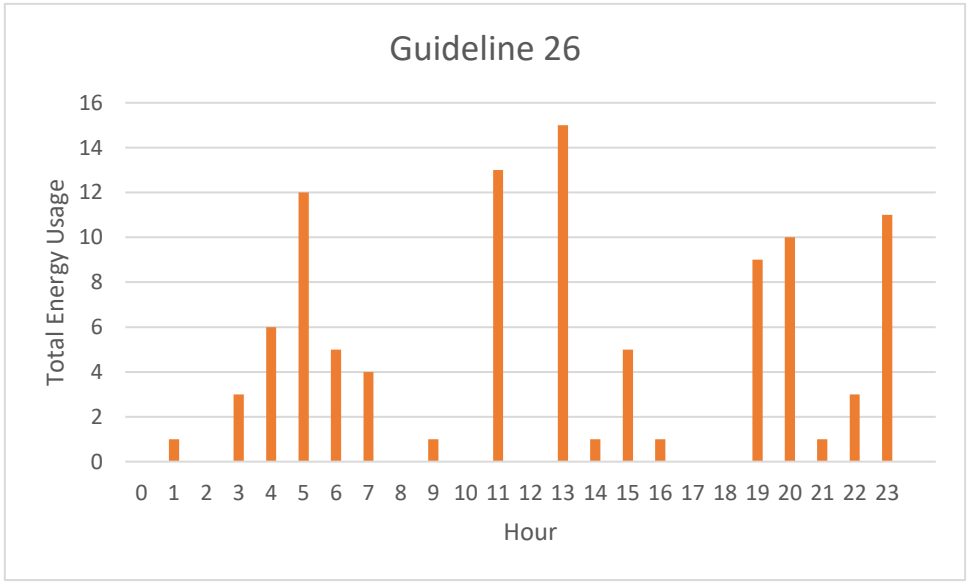

Guideline 17
Total Energy Usage
0
2
4
6
8
10
12
14
16
18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

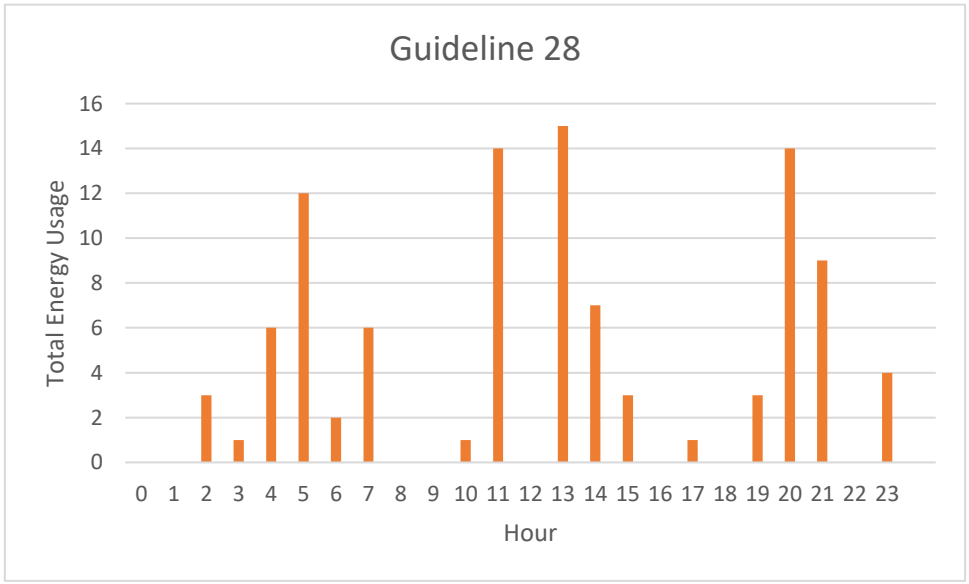

Guideline 18
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

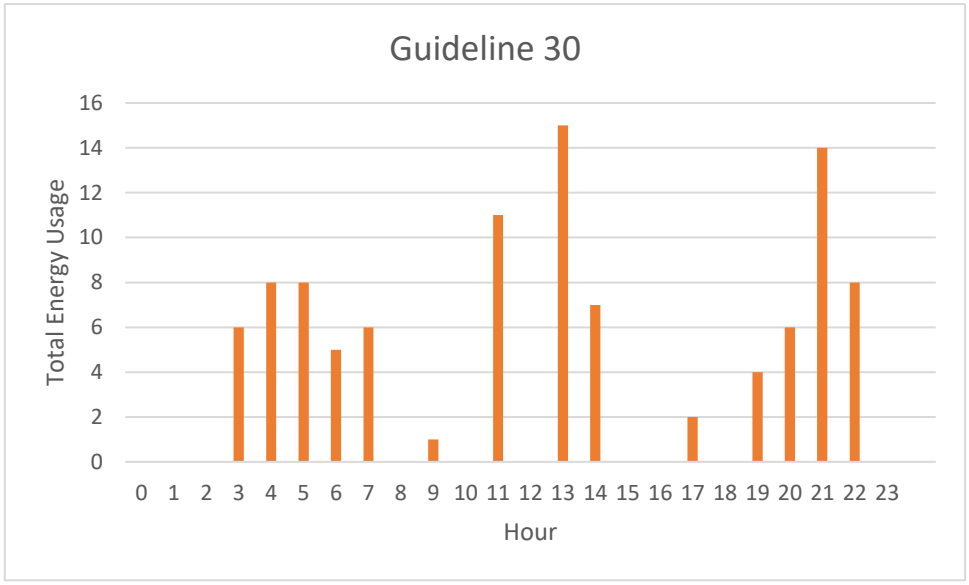

Guideline 19
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

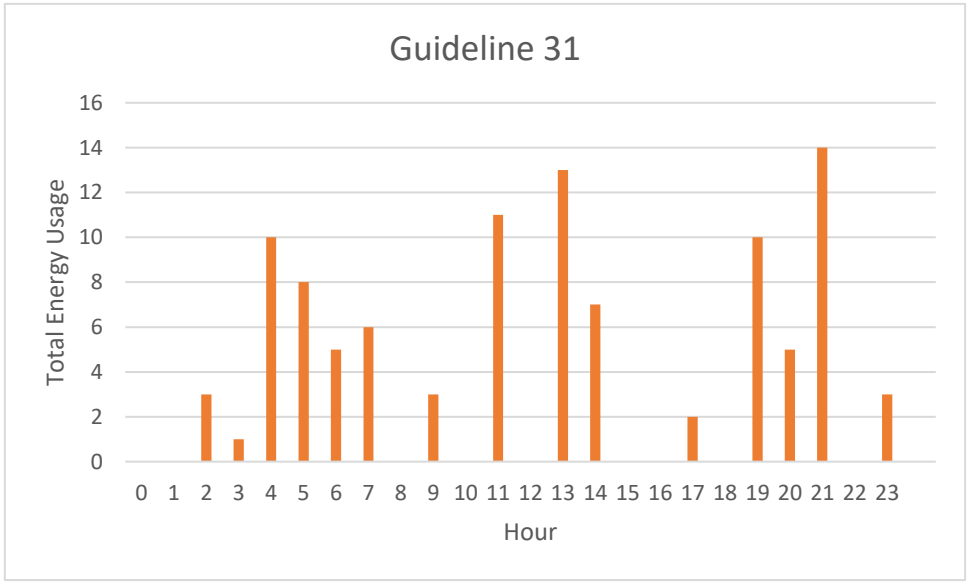

Guideline 20
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

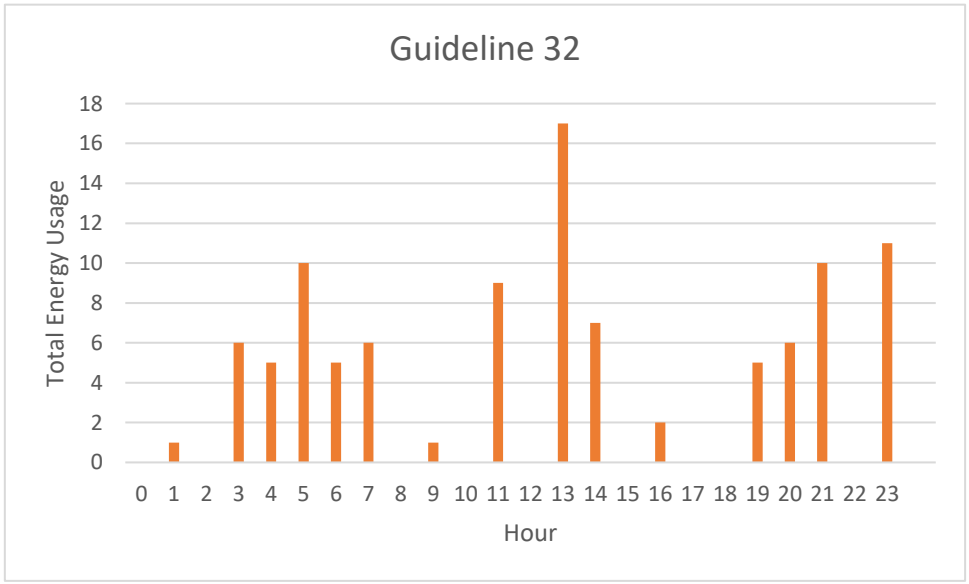

Guideline 22
Total Energy Usage
0 2 4 6 8 10 12 14 16 18 20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

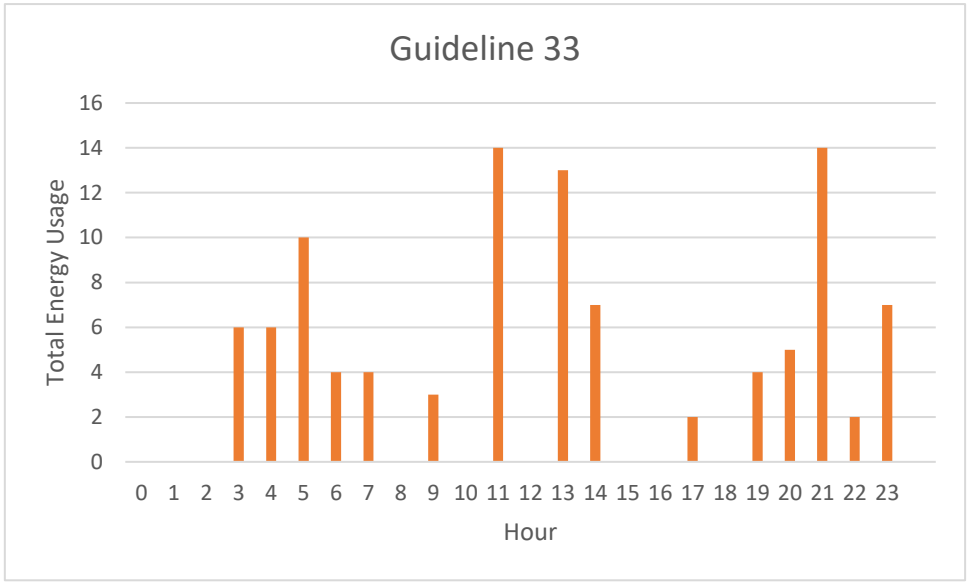

Guideline 26
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

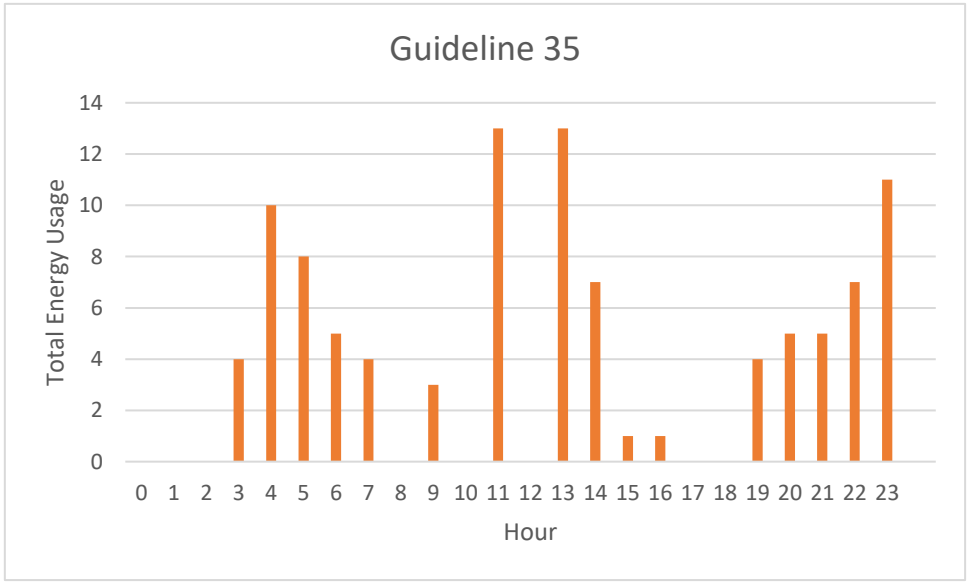

Guideline 28
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

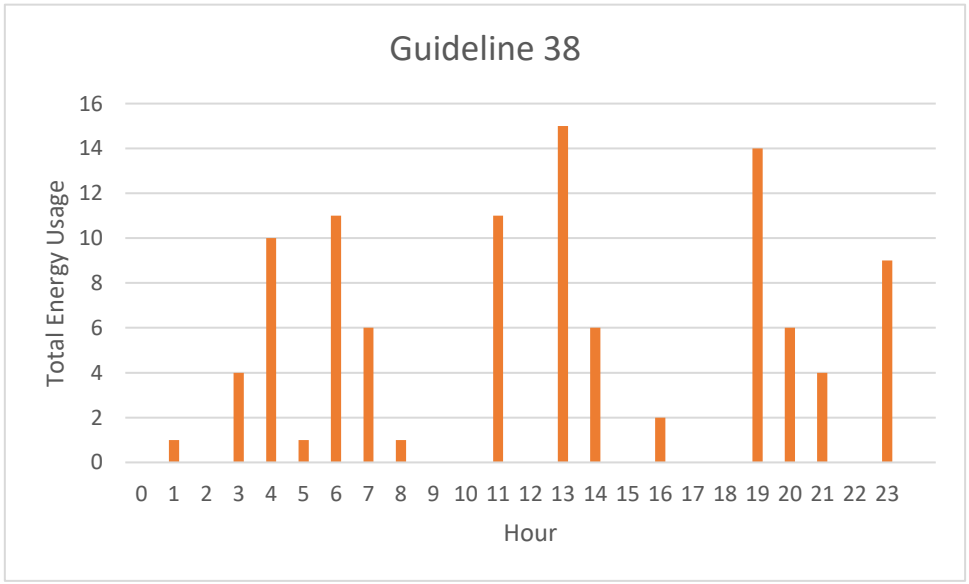

Guideline 30
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

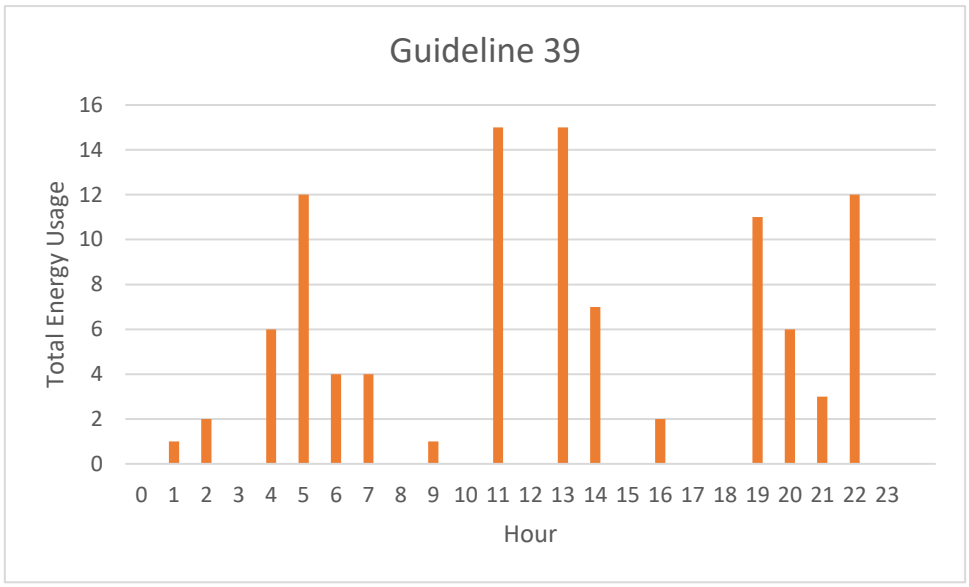

Guideline 31
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

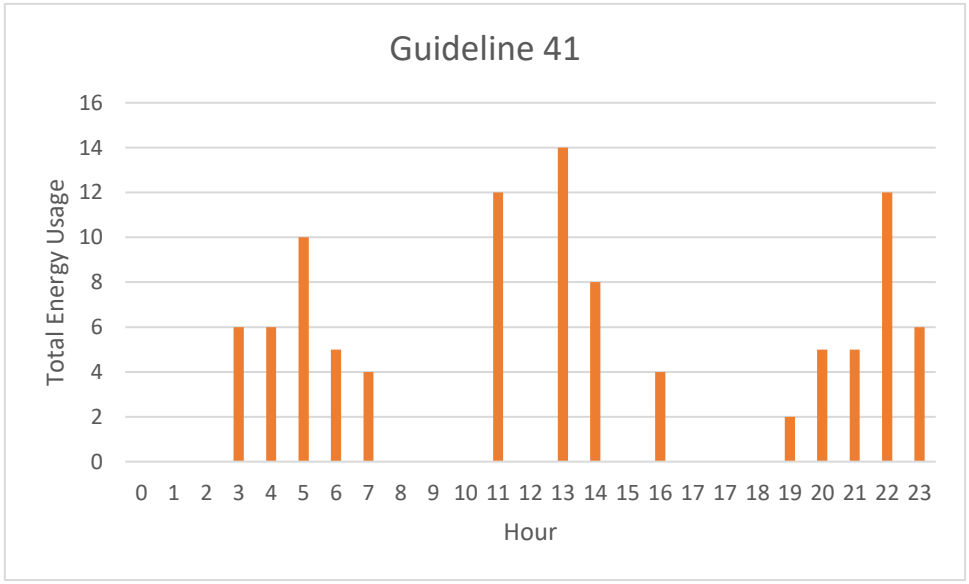

Guideline 32
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

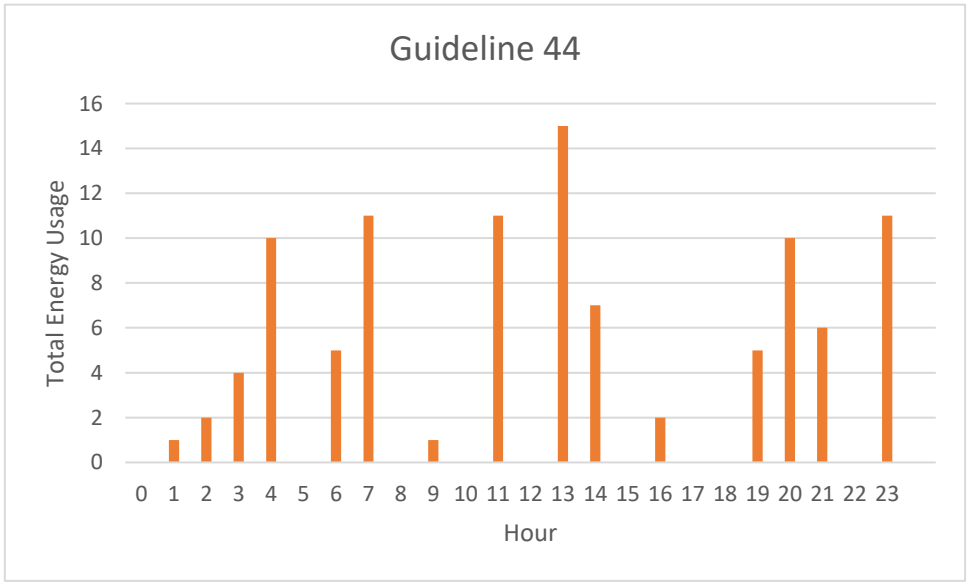

Guideline 33
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

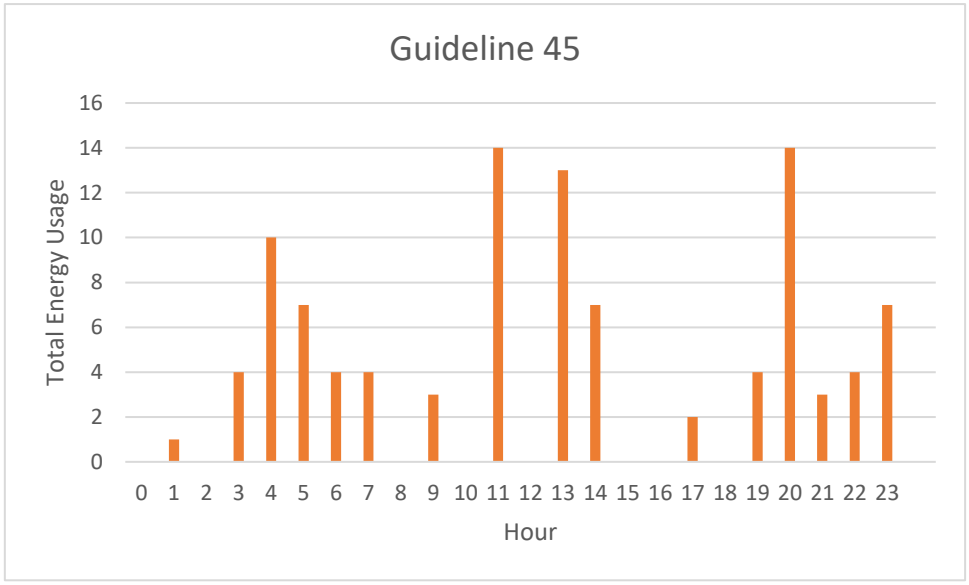

Guideline 35
Total Energy Usage
0 2 4 6 8 10 12 14
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

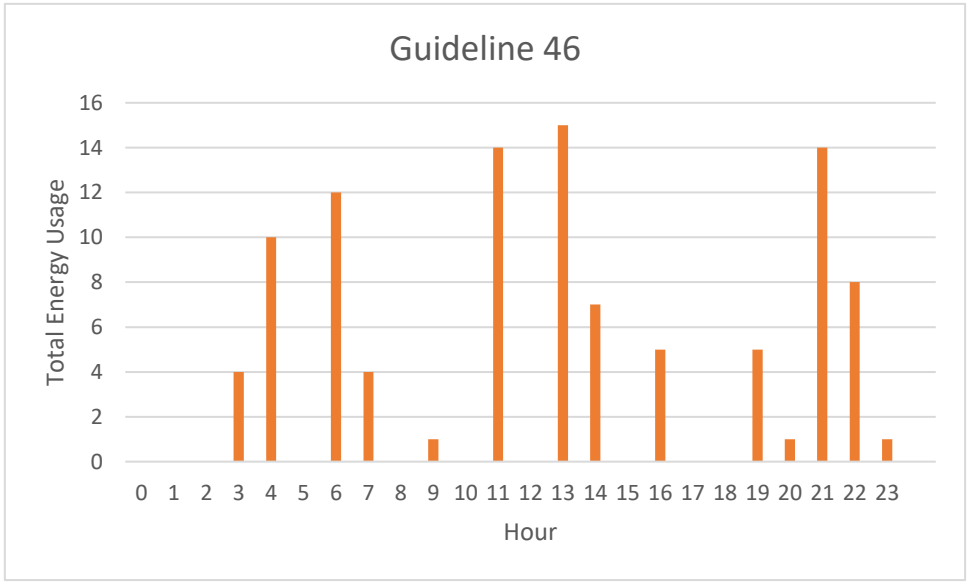

Guideline 38
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

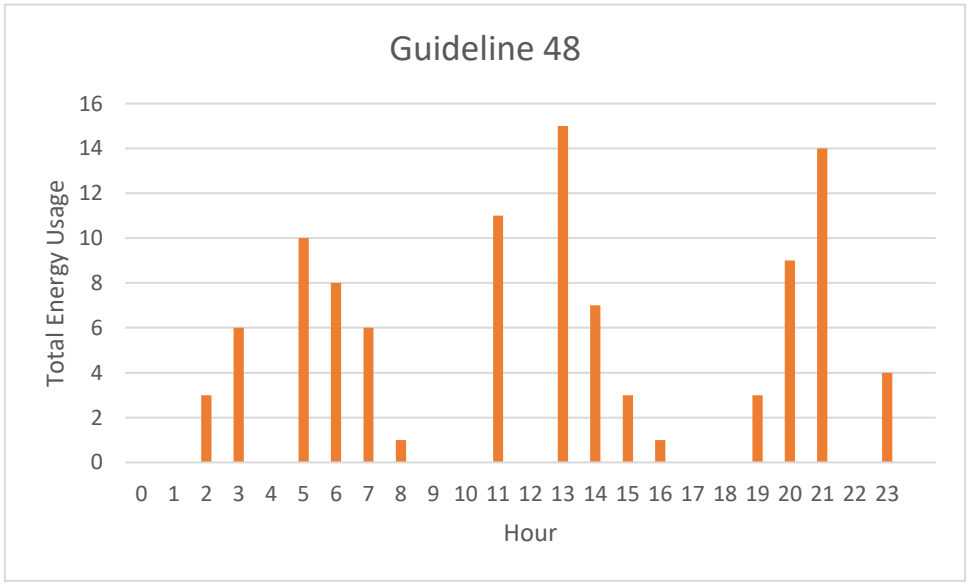

Guideline 39
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

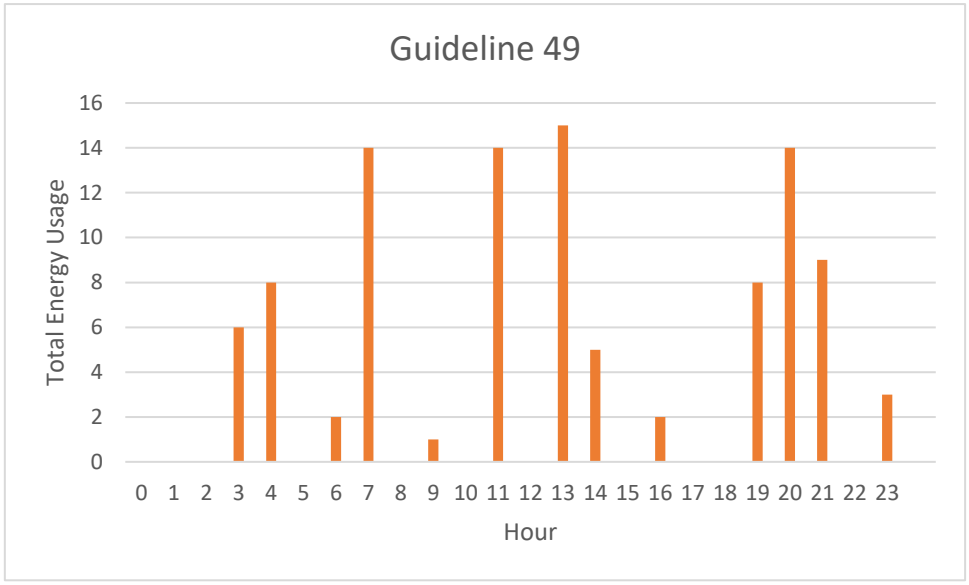

Guideline 41
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 17 18 19 20 21 22 23
Hour

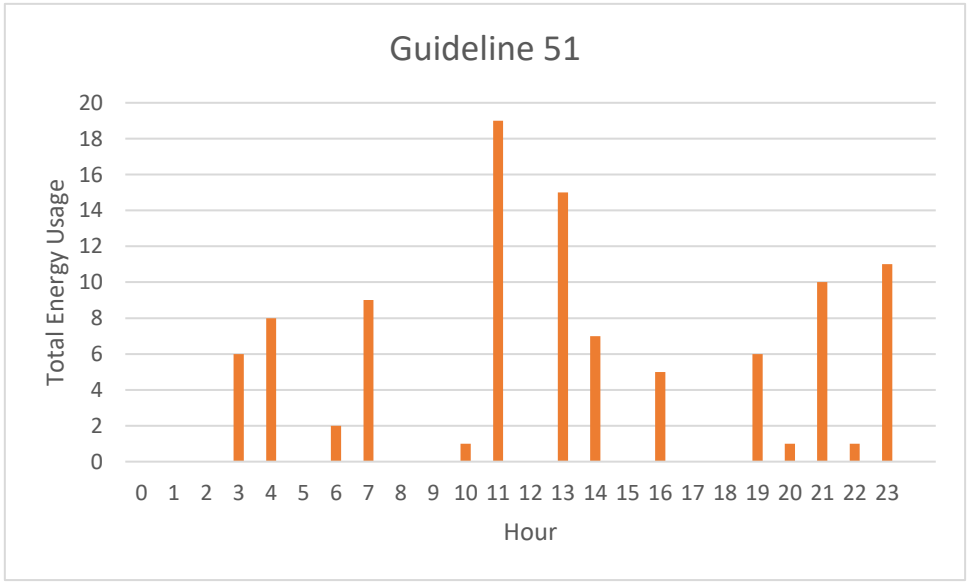

Guideline 44
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

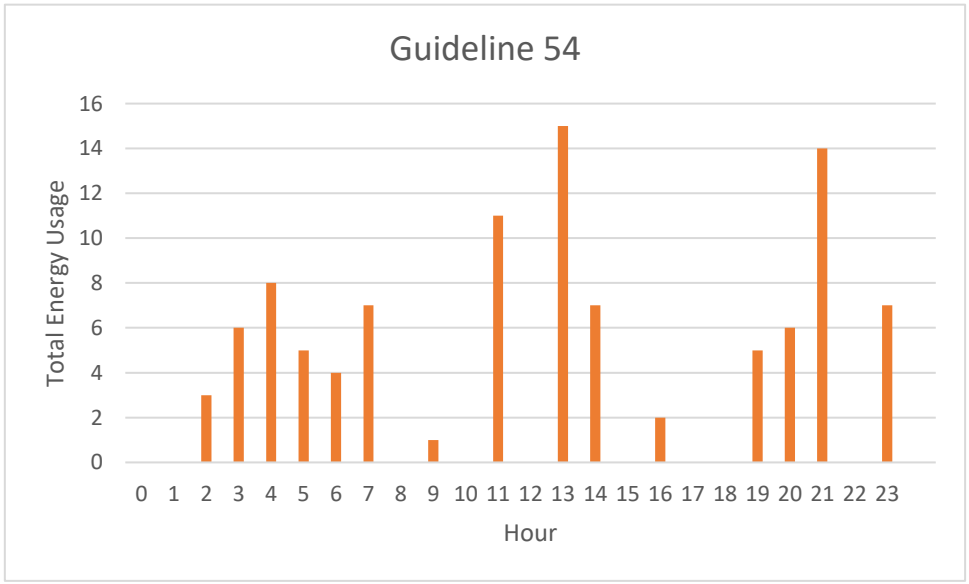

Guideline 45
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

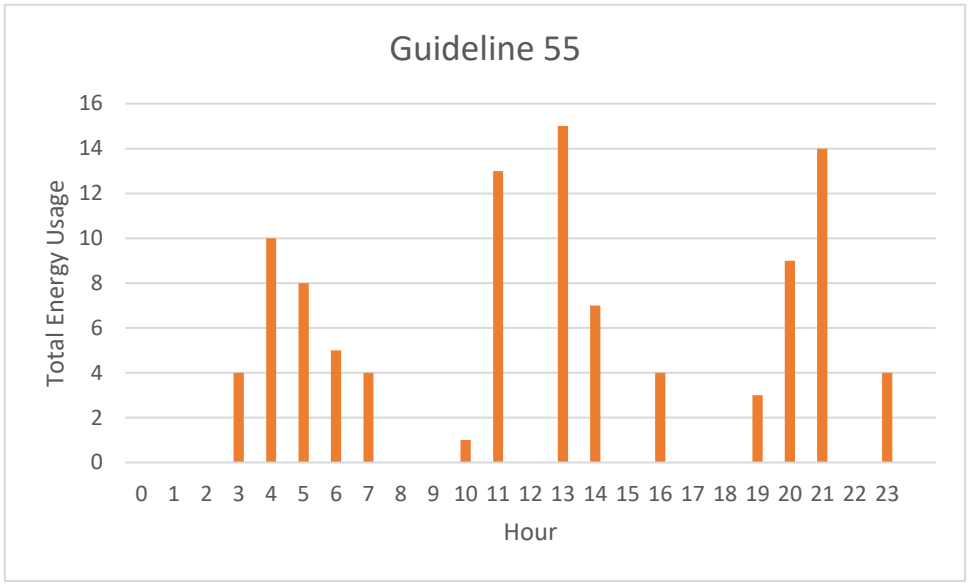

Guideline 46
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

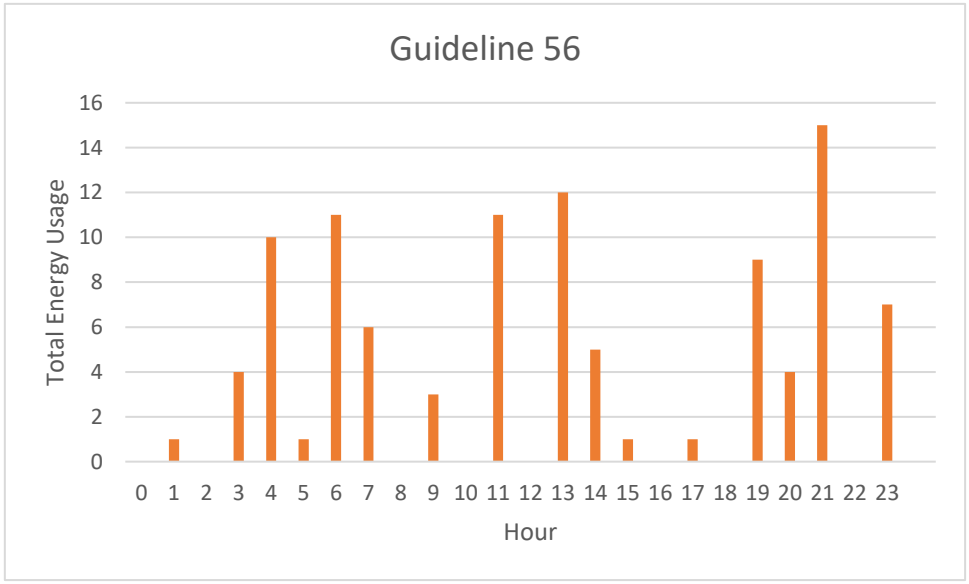

Guideline 48
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

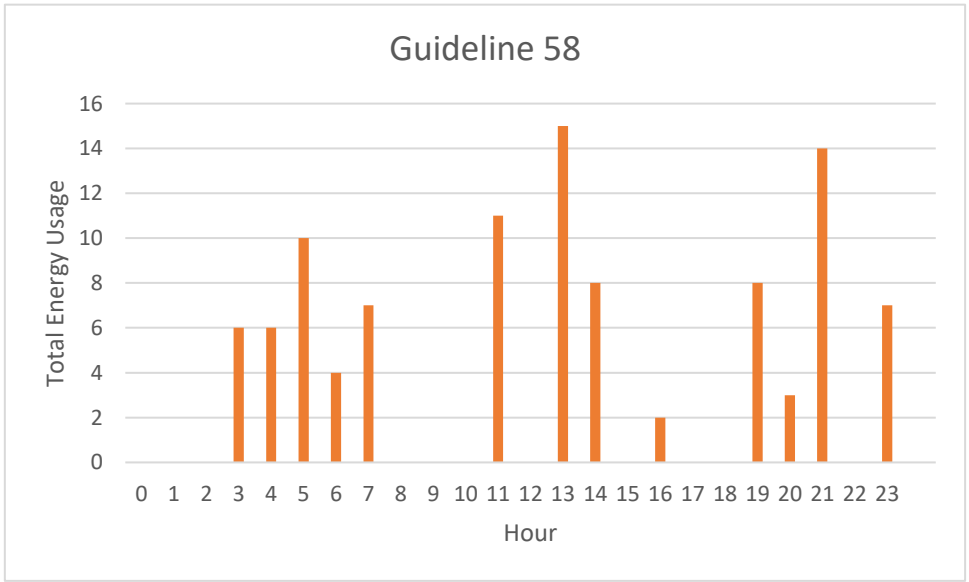

Guideline 49
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

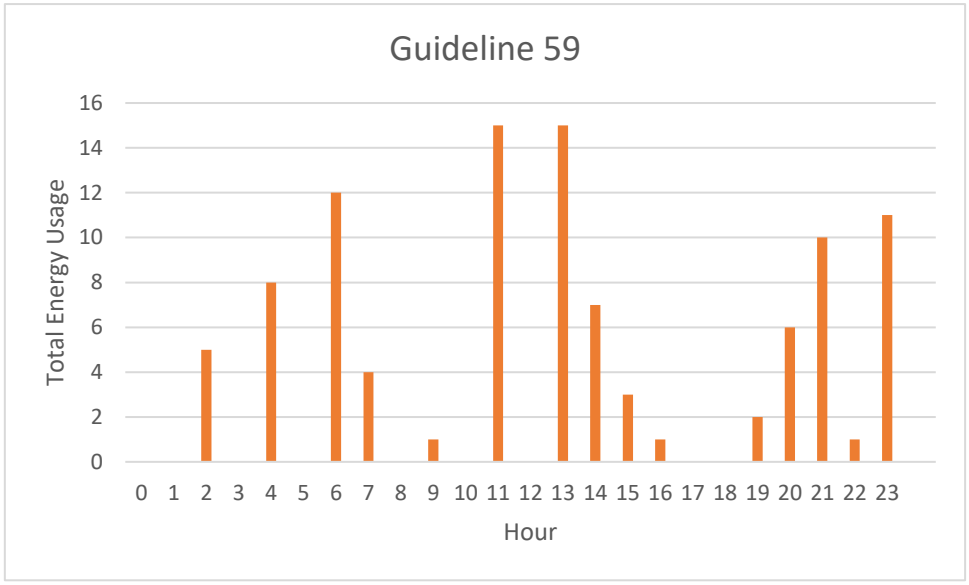

Guideline 51
Total Energy Usage
0
2
4
6
8
10
12
14
16
18
20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

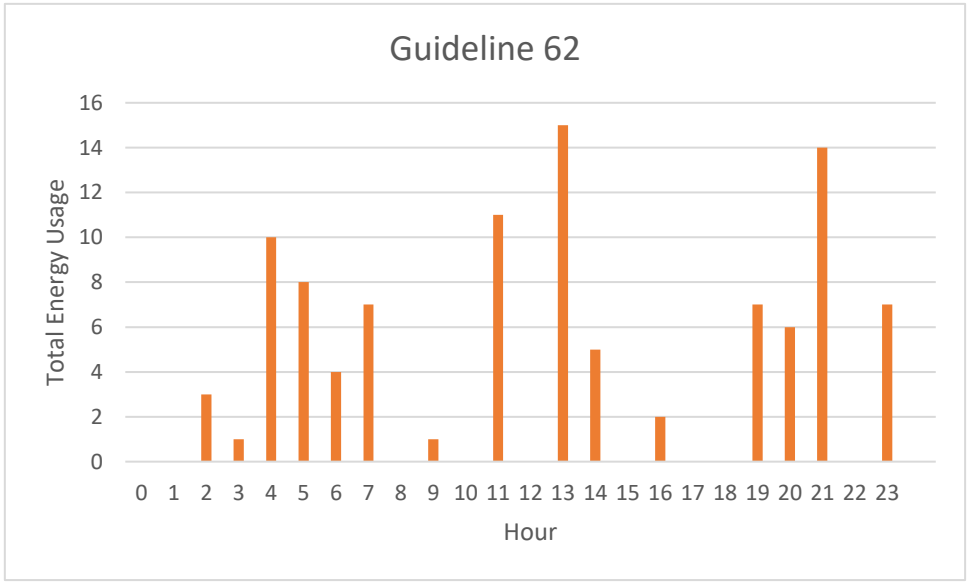

Guideline 54
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

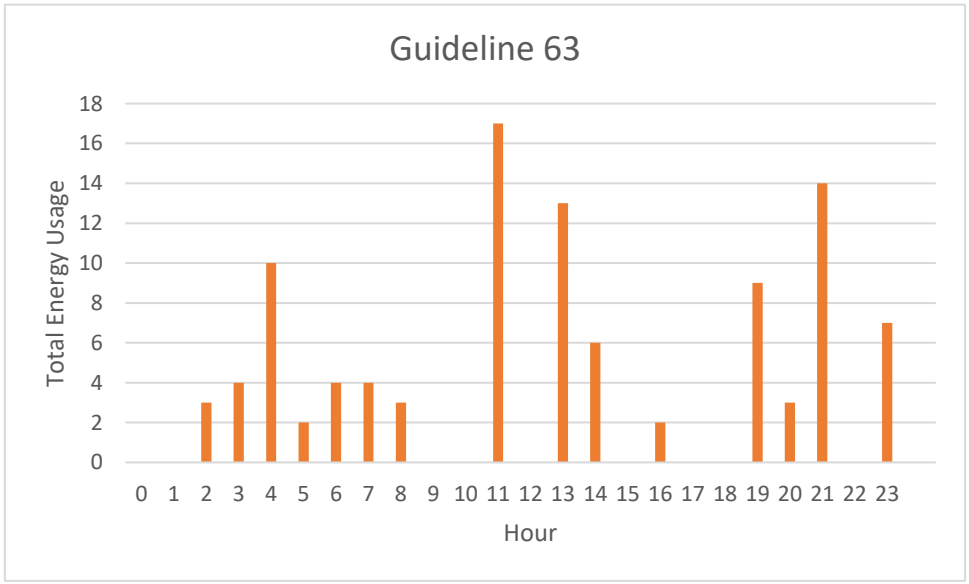

Guideline 55
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

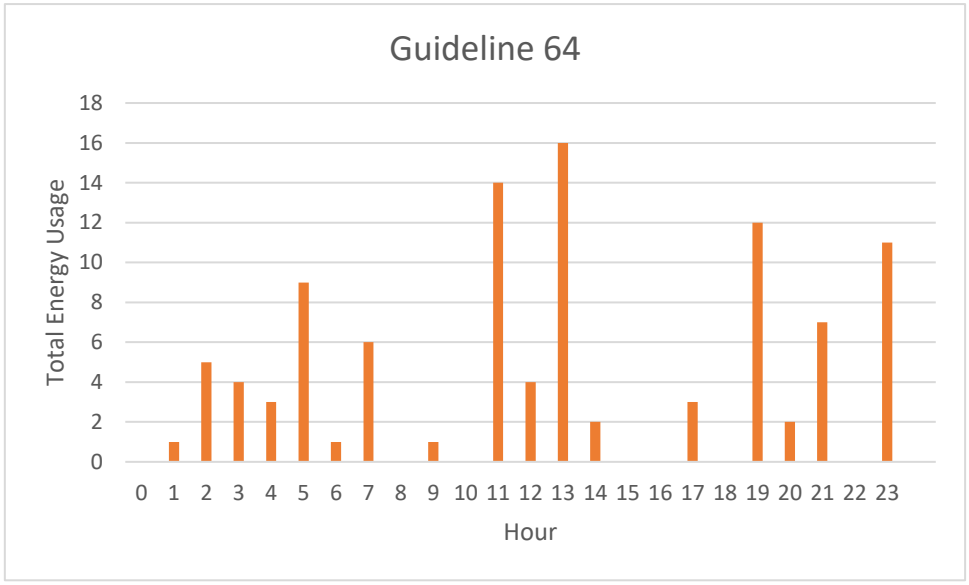

Guideline 56
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

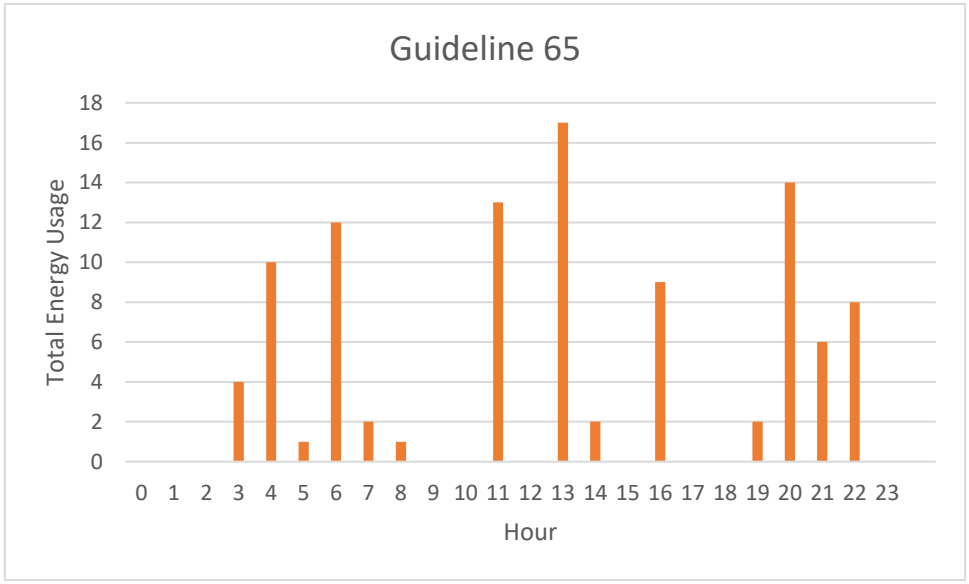

Guideline 58
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

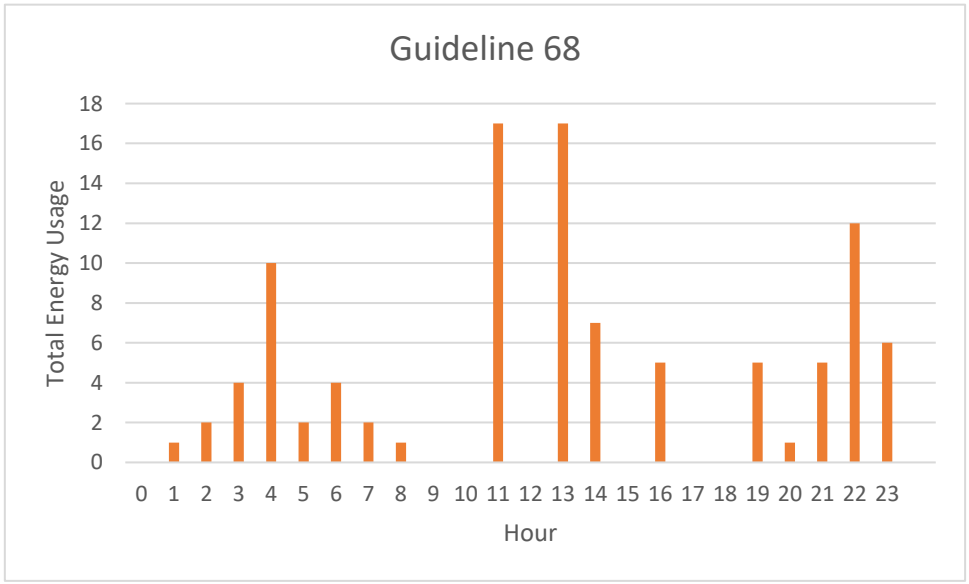

Guideline 59
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

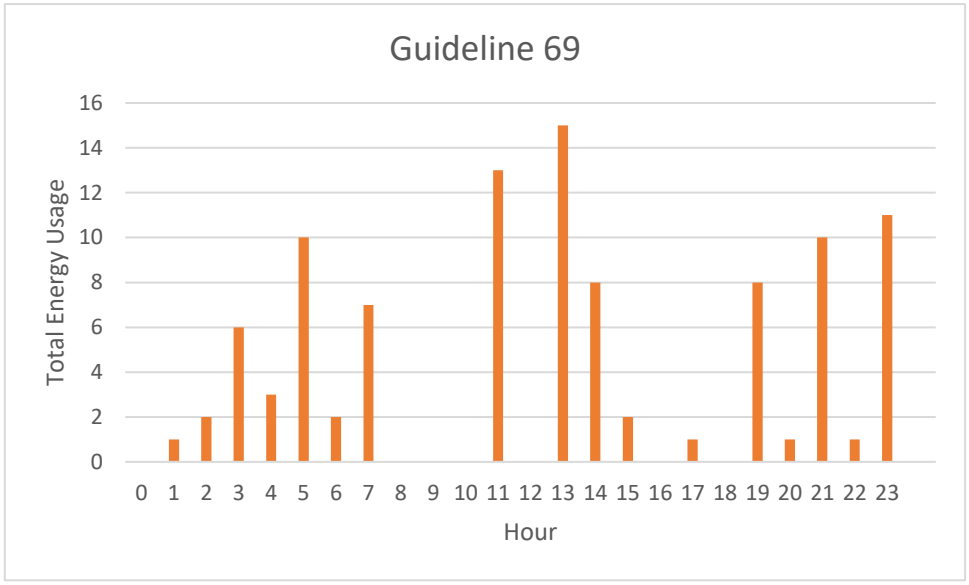

Guideline 62
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

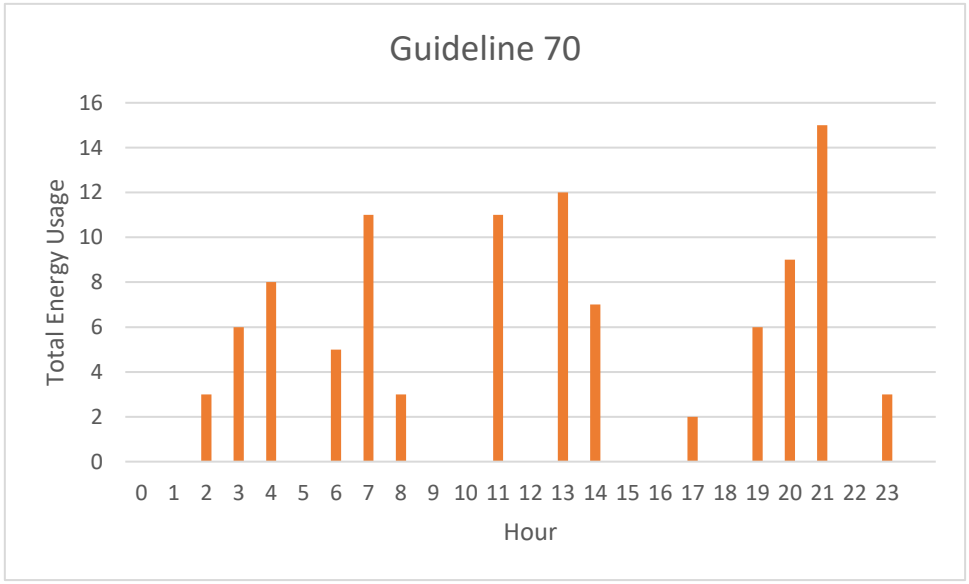

Guideline 63
Total Energy Usage
0
2
4
6
8
10
12
14
16
18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

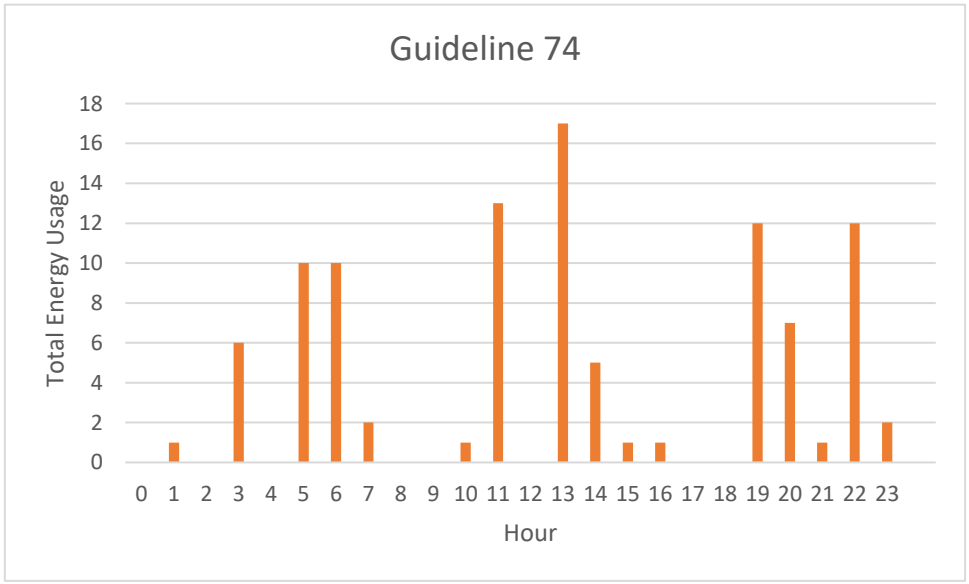

Guideline 64
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

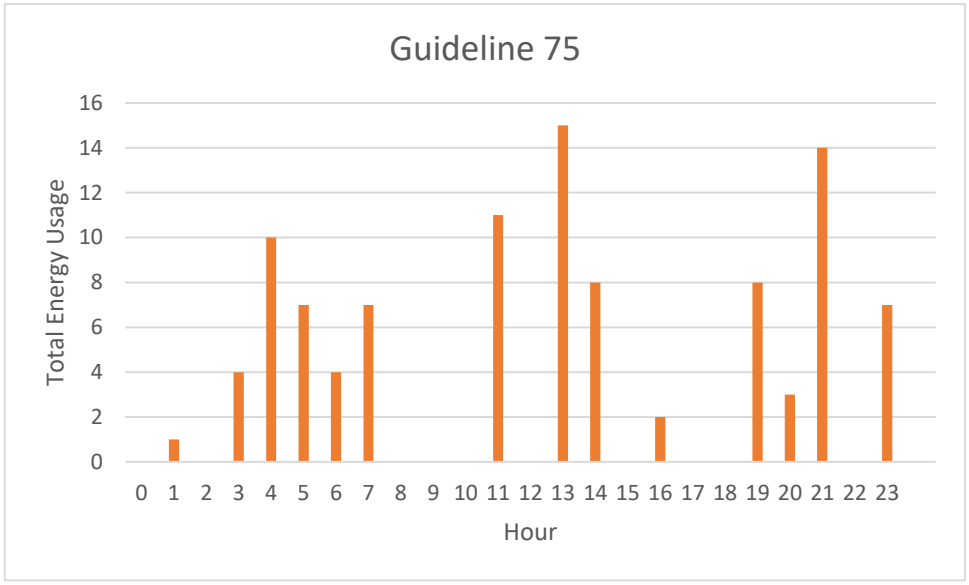

Guideline 65
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

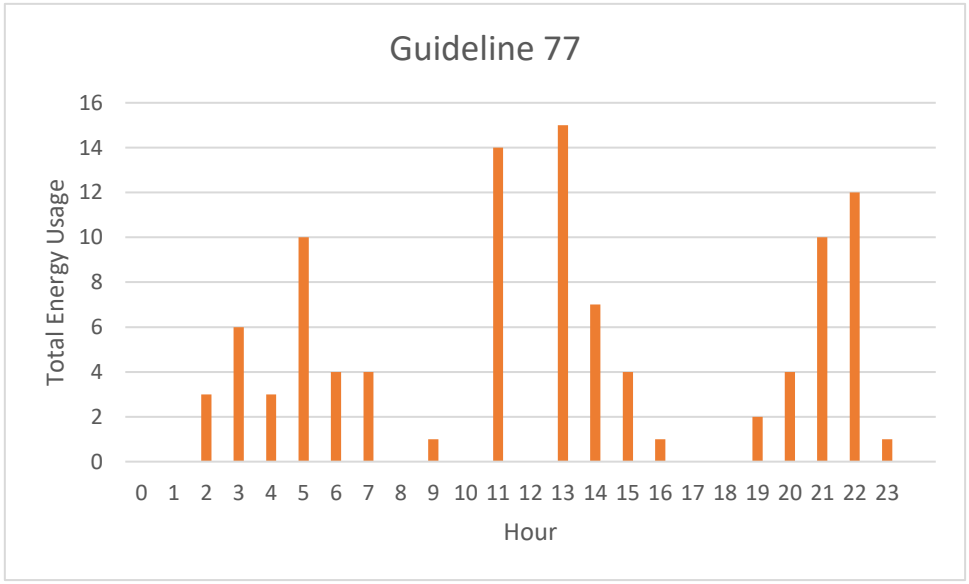

Guideline 68
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

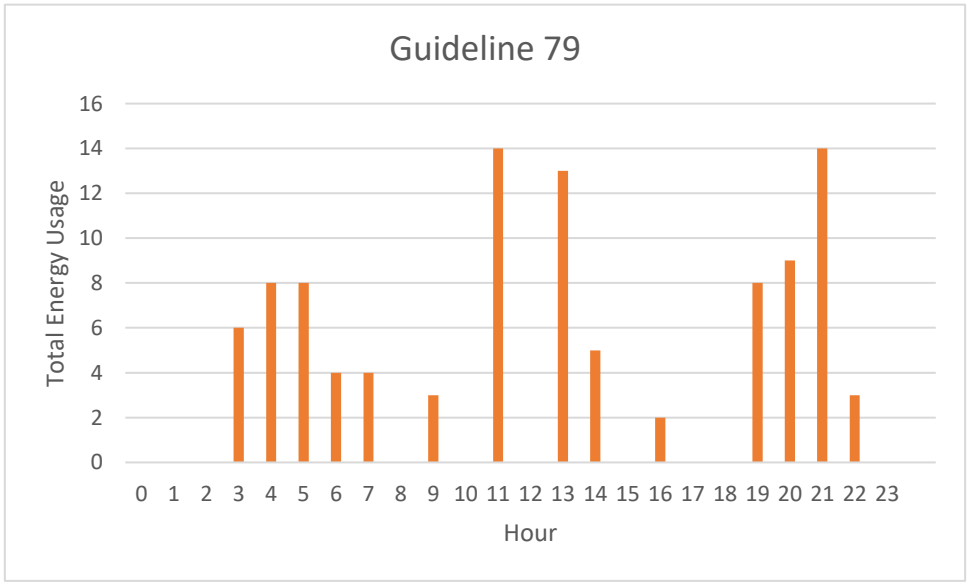

Guideline 69
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

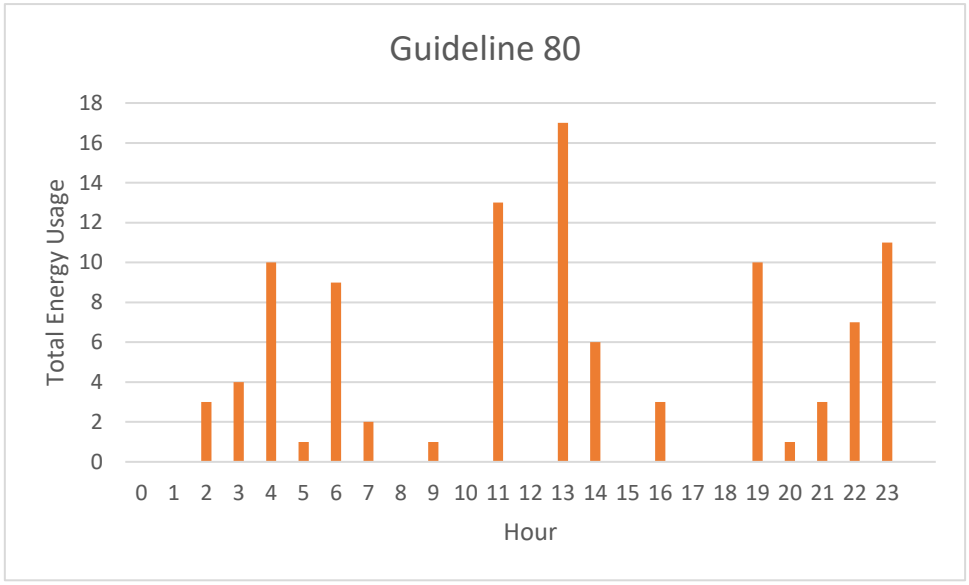

Guideline 70
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

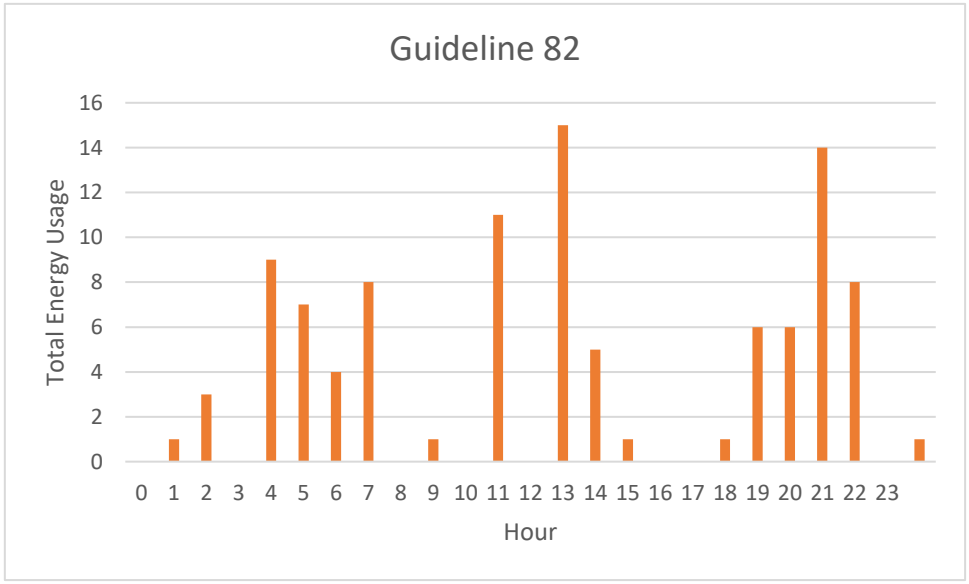

Guideline 74
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

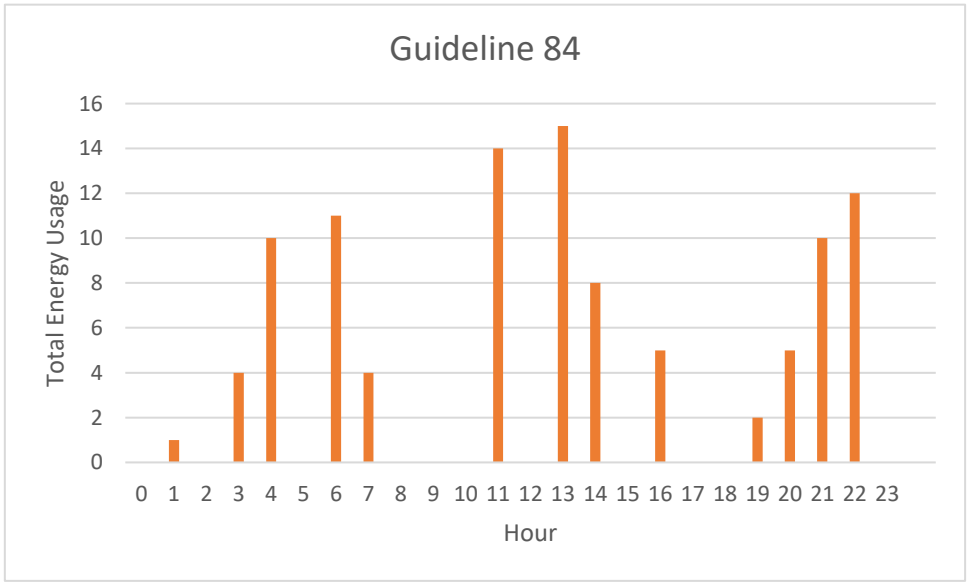

Guideline 75
Total Energy Usage
16
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

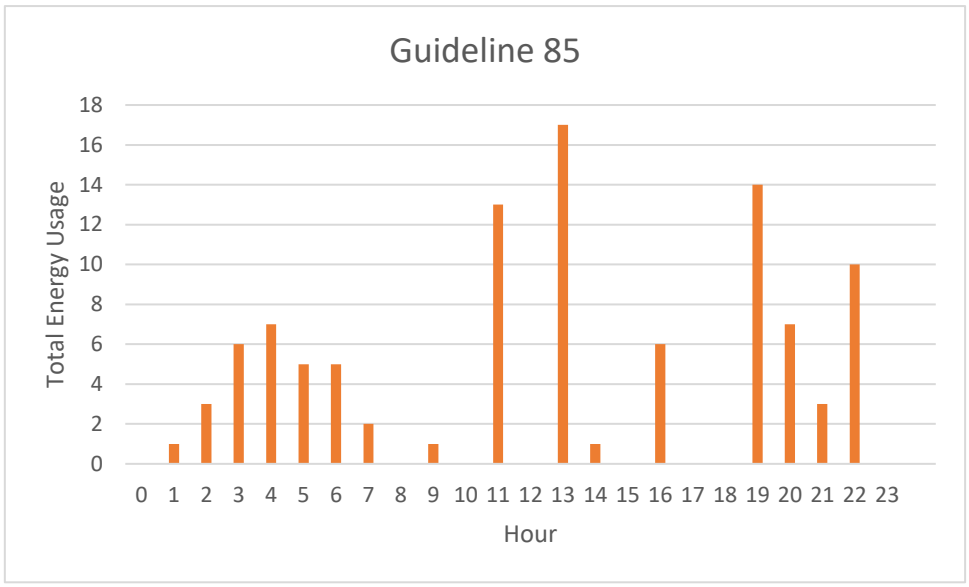

Guideline 77
Total Energy Usage
16
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

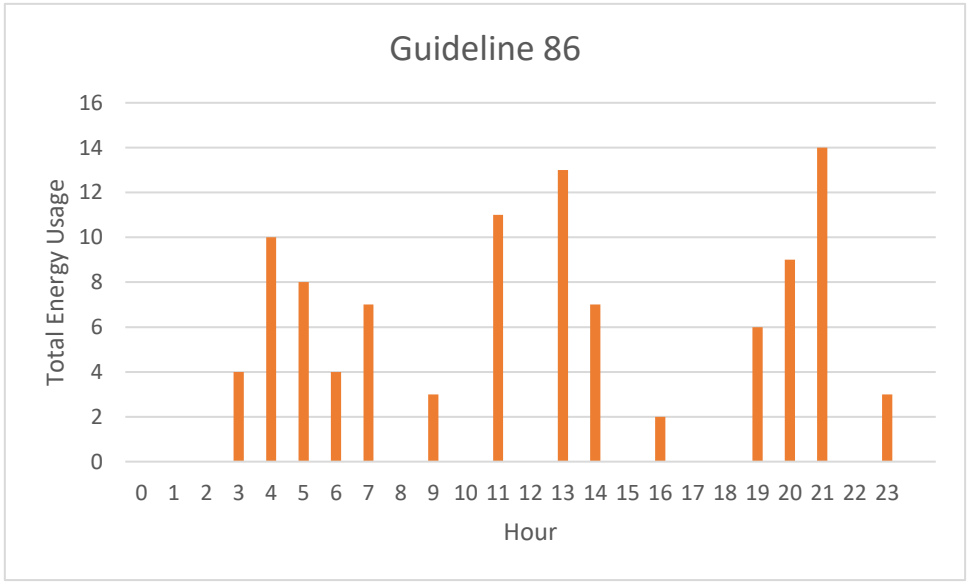

Guideline 79
Total Energy Usage
16
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

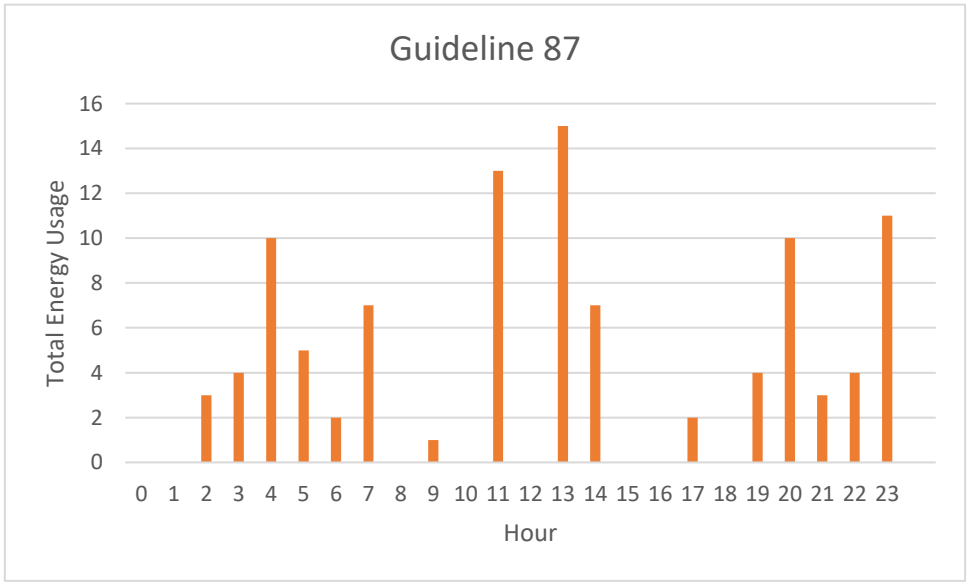

Guideline 80
Total Energy Usage
0
2
4
6
8
10
12
14
16
18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

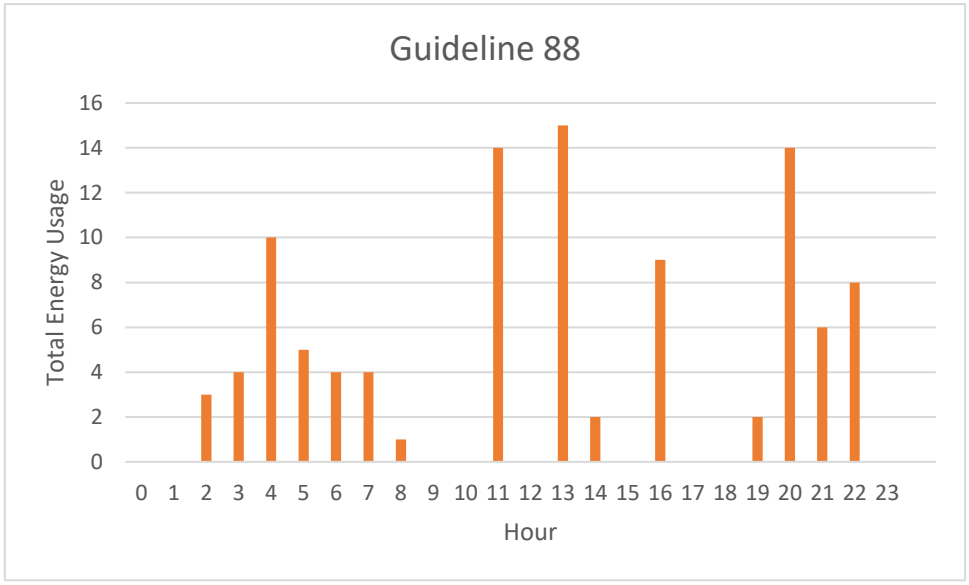

Guideline 82
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

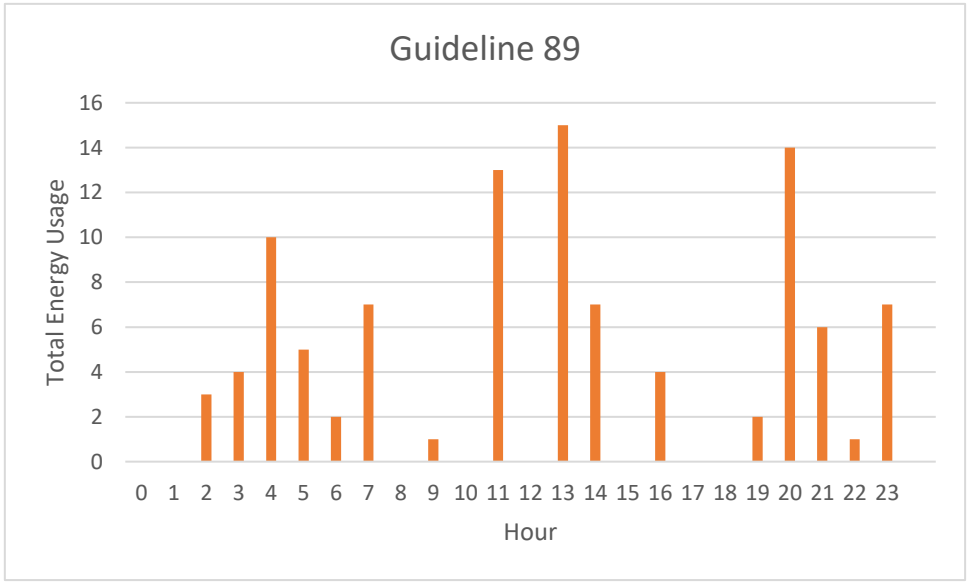

Guideline 84
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

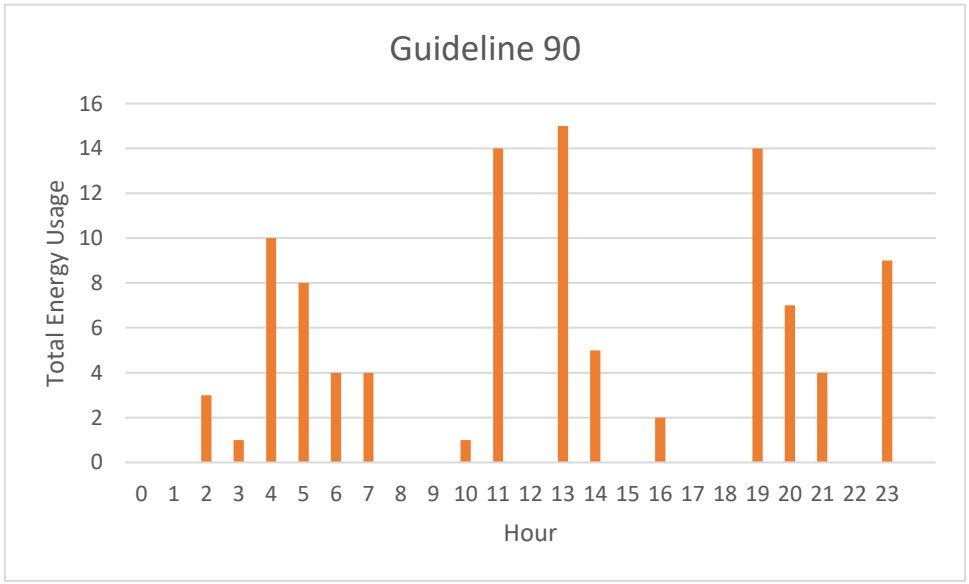

Guideline 85
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

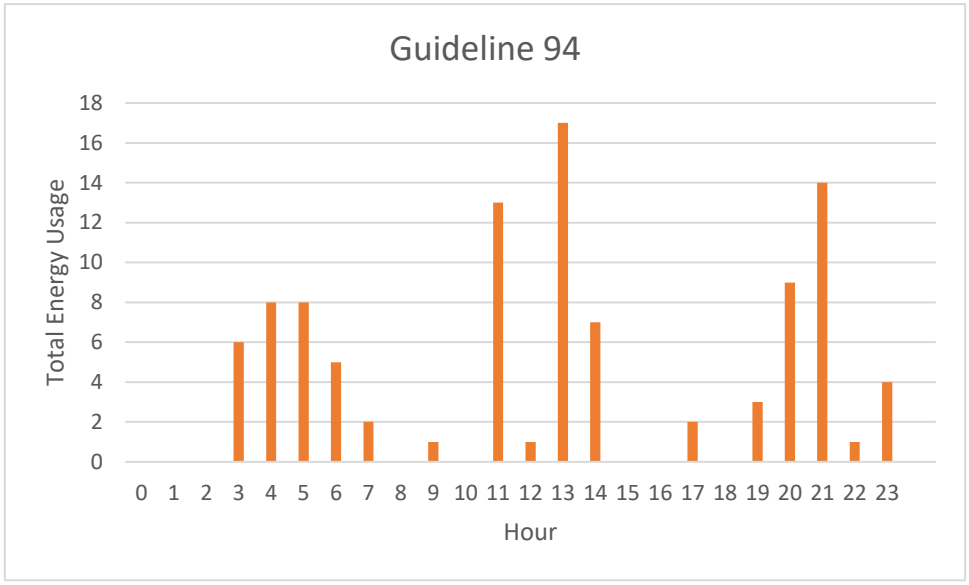

Guideline 86
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

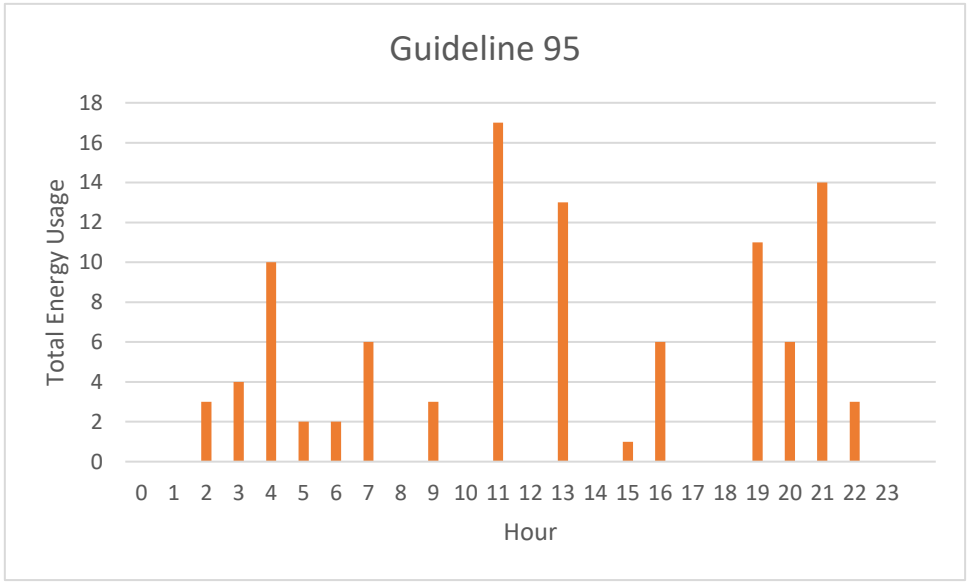

Guideline 87
Total Energy Usage
0 2 4 6 8 10 12 14 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

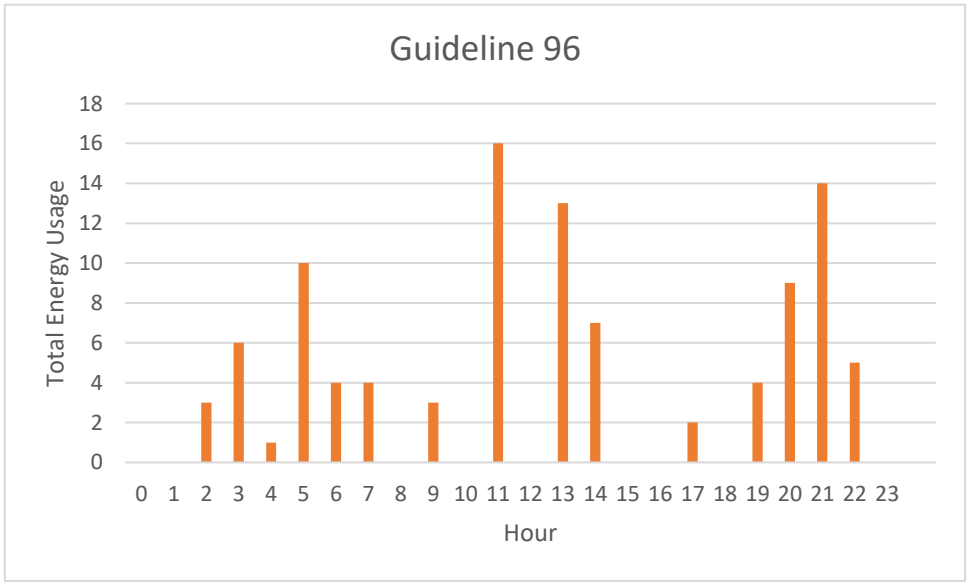

Guideline 88
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Guideline 89
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Guideline 90
Total Energy Usage
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Guideline 94
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Guideline 95
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Guideline 96
Total Energy Usage
0 2 4 6 8 10 12 14 16 18
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

Makefile

Using the CLI navigate to the root directory of the project and run the 'make' command. The first file that will run is 'sklearn_algorithm_comparison.py' which was used to compare the available algorithms and their accuracies so the best one could be chosen. The second file that will run is 'training_guideline_model.py' which trains the Gaussian Naïve Bayes classifier, runs it on the testing data and writes the results to the file 'TestingResults.txt'. The third file that will run is 'objective_function_generator.py' which generates the linear programming files for each user for each guidelines and places them in a subdirectory named 'lp'. A batch script is the run which using the CLI version of the 'LP_Solve' application to solve all of the linear programming files – for this to work, the 'lp_solve.exe' program needs to be in the same directory as all of the linear programming files. The batch script then combines all the different user .csv files into one .csv file per guideline. The guidelines .csv files are then convert into a Macro Enabled Excel Workbook (.xlsm) file. This was done so that the macros 'GraphMakerMacro.bas' and 'AxesTitles.bas' could be run on the on each workbook so that a graph of the combined energy usage could be obtained quickly and easily.

GitHub

All code and files produced were stored and maintained on a GitHub Repository named "COMP3217-Coursework-2": https://github.com/cargraham/COMP3217-Coursework-2

References

[1] N. Chakrabarty, "Implementation of Gaussian Naive Bayes in Python from scratch," *Hackernoon*, Jan. 23, 2019. https://hackernoon.com/implementation-of-gaussian-naive-bayes-in-python-from-scratch-c4ea64e3944d (accessed Apr. 28, 2022).

[2] P. Majumder, "Gaussian Naive Bayes," *OpenGenus IQ*. https://iq.opengenus.org/gaussian-naive-bayes/#:~:text=Gaussian%20Naive%20Bayes%20supports%20continuous,(independent%20dimensions)%20between%20dimensions. (accessed Apr. 28, 2022).

[3] C. Y. Huang, C. Y. Lai, and K. T. Cheng, "Fundamentals of algorithms," *Electronic Design Automation*, pp. 173–234, Jan. 2009, doi: 10.1016/B978-0-12-374364-0.50011-4.