AMS 559
Smart Energy in the Information Age

Project #2

Pratik Mukund Velhal (112675099)

Project Overview:

This project incorporates evaluation of performance of different electricity provisioning algorithms based on the demand data and predictions of three houses. We are using data for 3 houses that contain the load demand (in kW) for each 30-minute interval over the first 14 days of November (Nov. 1 to Nov. 14). We are using provisioning algorithms like online gradient descent, receding horizon control, commitment horizon control to optimally provision electricity for each house.

We are using the following objective function to measure performance:

$$\sum_{t=1}^{T} p(t)x(t) + a * \max\{0, y(t) - x(t)\} + b|x(t) - x(t-1)|$$

Values of constants are as follows:
Price (p) = 0.4$/kwH = (0.4/2) $ / kW 30 min = 0.2$ / KW 30 min = 0.2
a = b = 4$/kwH = 2$/kw 30 min = 2

Implementation:

All the required tasks have been implemented as follows:

1. Solving the offline optimization problems using CVX in Python

We have performed both static and dynamic offline optimization for all the three users (B, C and F) respectively.

1.1. *Static Offline Optimization*:

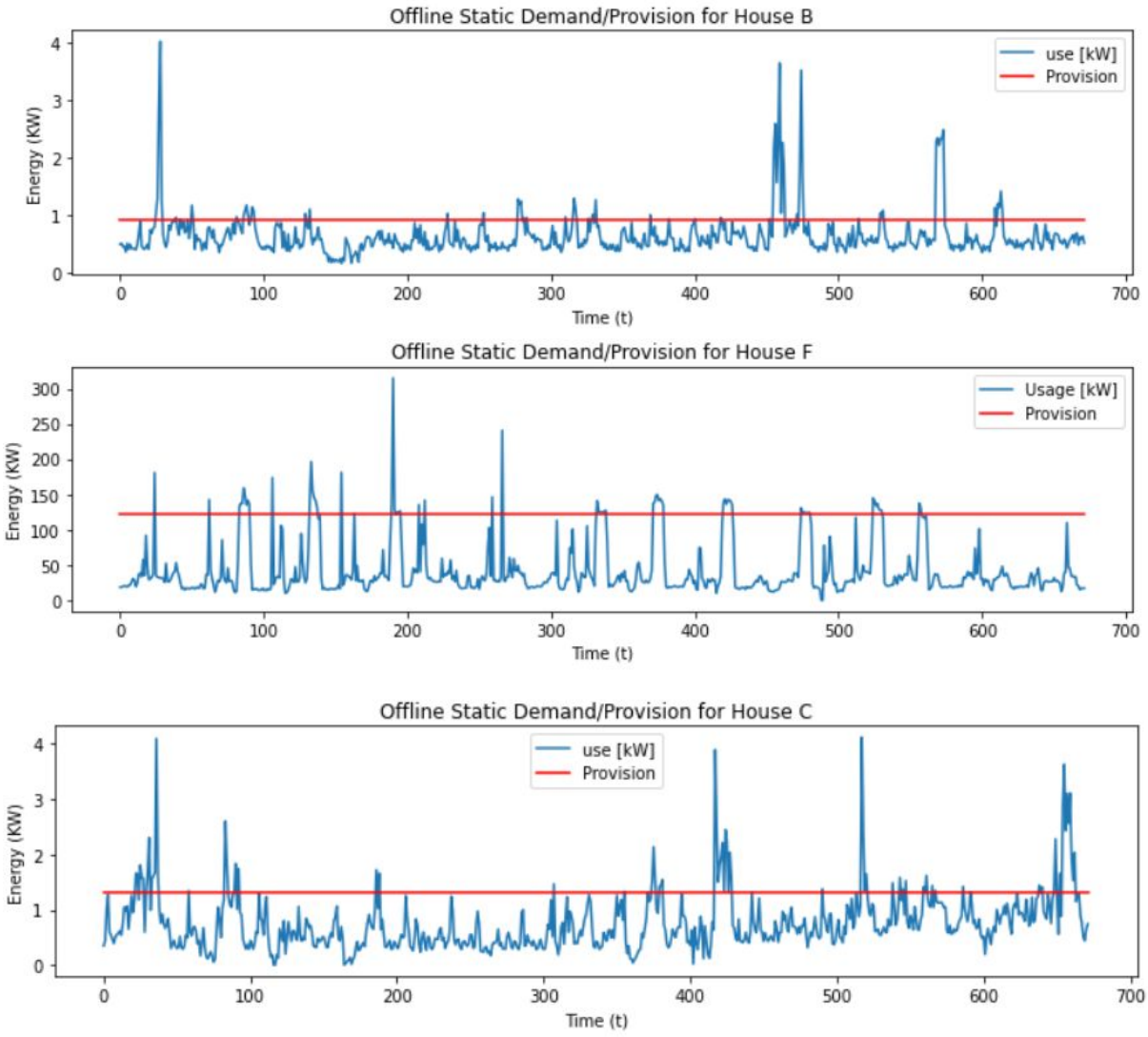
For static offline optimization, the optimal value is calculated as:

```
localcost += (price*x + a*maximum(0,y[t]-x))
```

We are using cvxpy to minimize this objective function for a given x. For this value of x, we are then finding the optimal value of cost.
We get the following values of optimal cost using static offline optimization:

House B	House C	House F
193.024	257.280	19154.158

The demand/provision for the three users using Static offline optimization are as follows:



1.2. *Dynamic Offline Optimization*:

For dynamic offline optimization, we define the objective function as

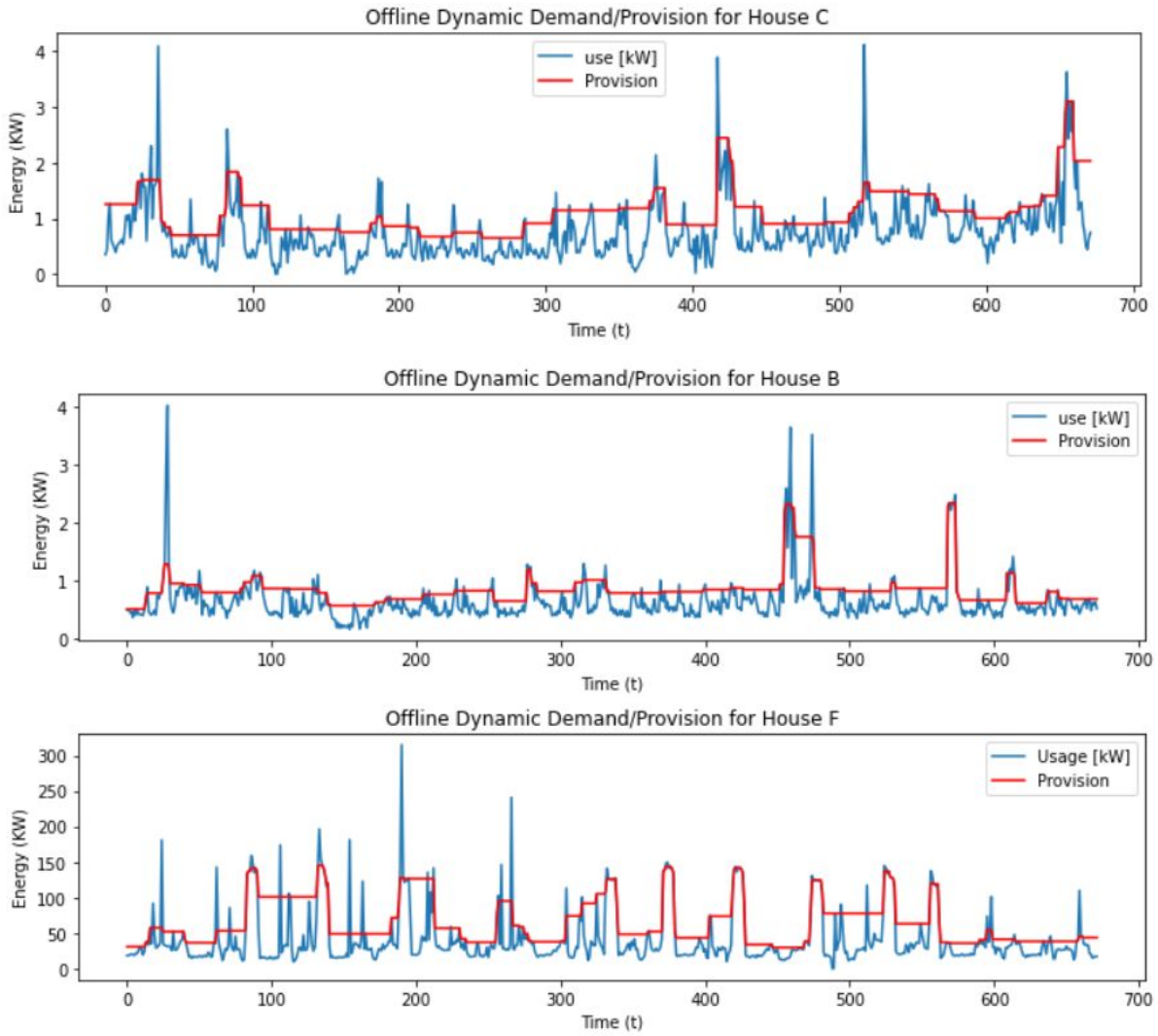
```
localcost += (price*x[t] + a*maximum(0,y[t]-x[t]) + b*abs(x[t]-x[t-1]))
```

We are using cvxpy to minimize this objective function for a given x. For this value of x, we are then finding the optimal value of cost.

We get the following values of optimal cost using dynamic offline optimization:

House B	House C	House F
160.471	209.799	14744.553

The demand/provision for the three users using dynamic offline optimization are as follows:



2. Online Algorithms:

We have used Online Gradient Descent (OGD), Receding Horizon Control (RHC) and Commitment Horizon Control (CHC) in this section to find the optimal provision.

2.1. *Online Gradient Descent (OGD):*

In this algorithm, our objective function is written as

x[t+1] = x[t] - stepsize*(gradient)

where at each iteration, we calculate the value of x using the value of x in the previous iteration and gradient, which is a partial differentiation of objective function by x(t). We have run this algorithm with multiple values of both static and dynamic step-sizes, and chosen the value of step size that gives the most optimal solution.

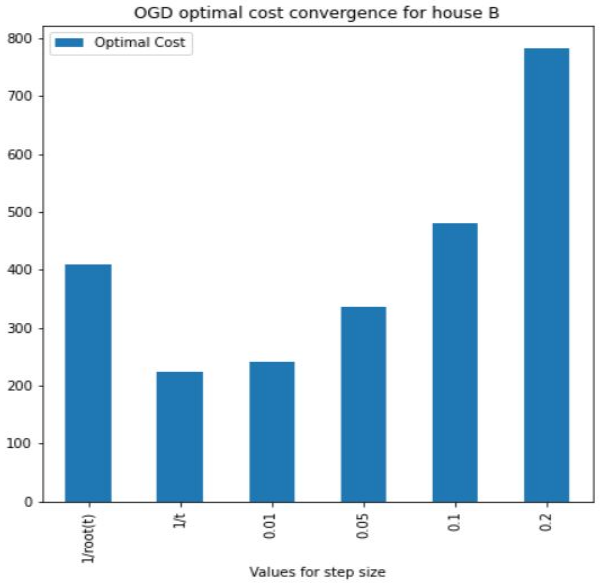
We get the following values of optimal cost using dynamic offline optimization:

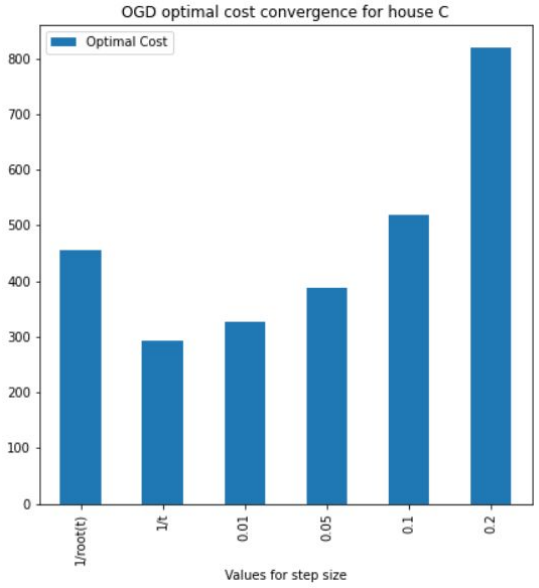
House	Step Sizes					
	1/root(t)	1/t	0.01	0.02	0.1	0.2
B	409.486	223.688	241.398	335.071	480.256	782.376
C	455.335	293.986	326.133	388.721	519.761	819.076

House	Step Sizes			
	1/t	1/root(t)	1	5
F	44782.387	27506.092	23832.682	31830.718

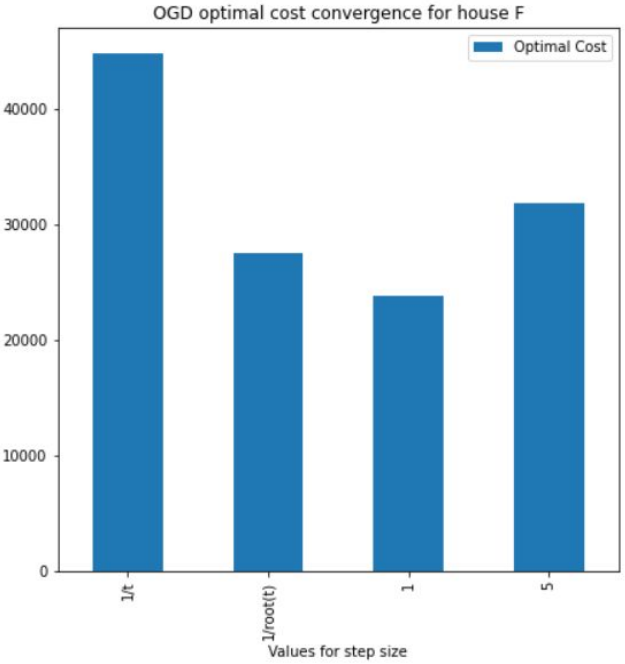
For houses B and C, we tried using 6 different values of step-sizes. Among them, 1/root(t) and 1/t were dynamic step-size heuristics where t is the iteration number. We also tried using static step-sizes of 0.01, 0.02, 0.1, 0.2 respectively. Of all the step-sizes, the most optimal solution was found for step size = “1/t” for both B and C.

We have also visualized the effect of step-size on value of objective function as follows:

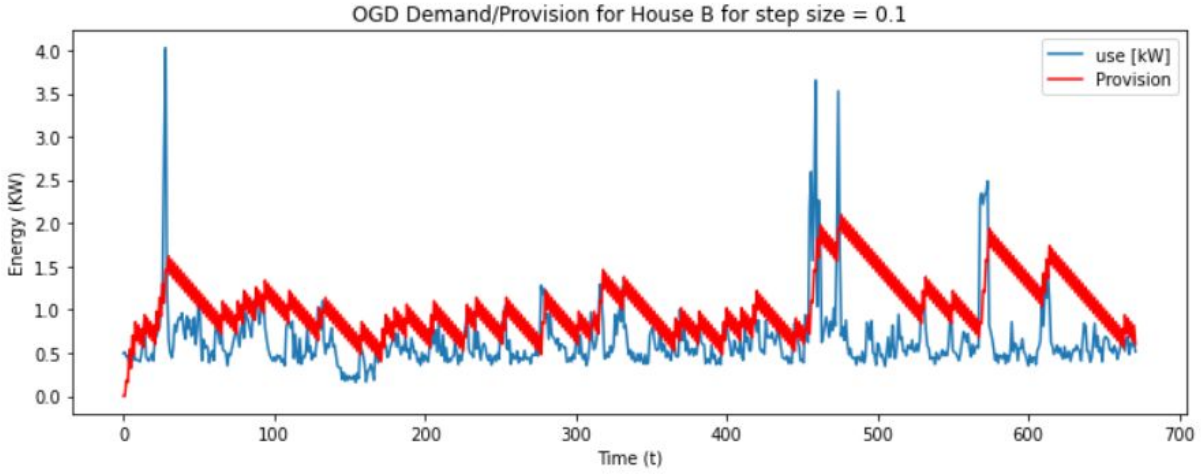


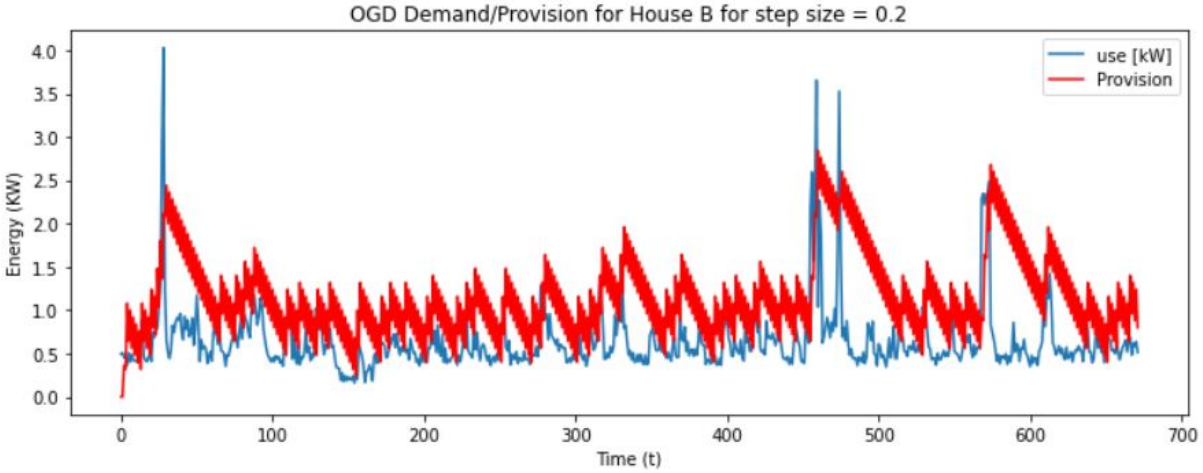


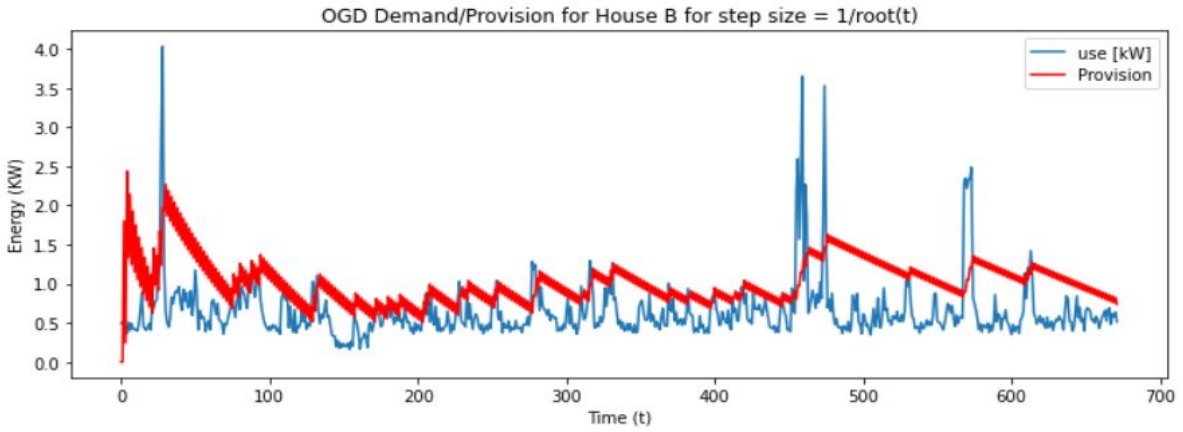
For houses F, we tried using 4 different values of step-sizes, ie. 1/root(t), 1/t, 1 and 5. Of all the step-sizes, the most optimal solution was found for step size = “1”. Similarly, the following graph shows effect of step-size on the optimal value:

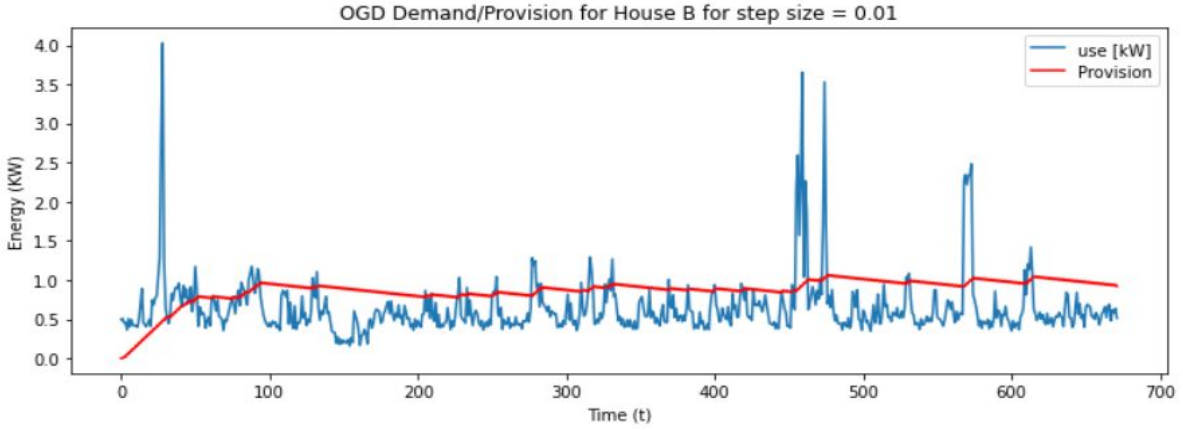


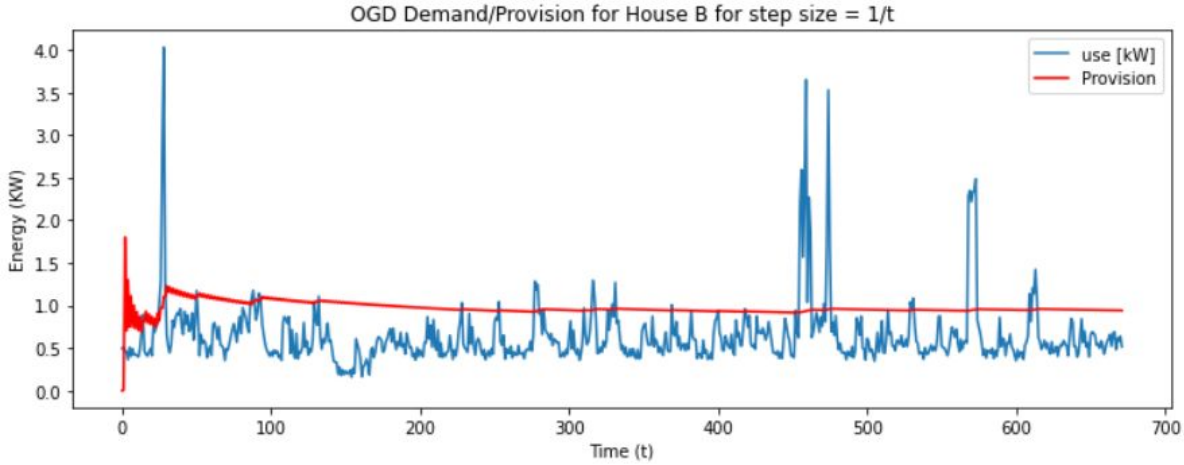
The demand/provision for the three users using Online gradient descent are as follows:

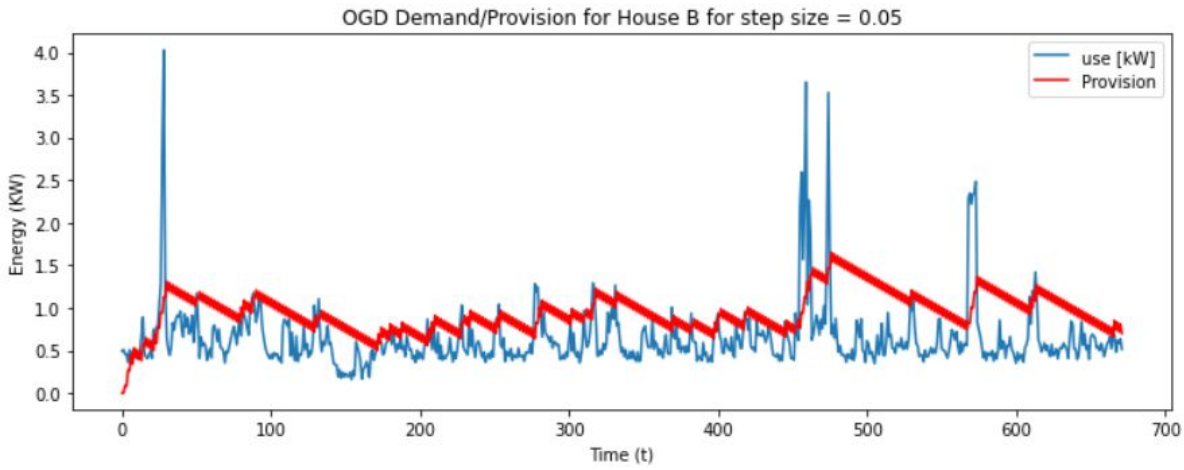


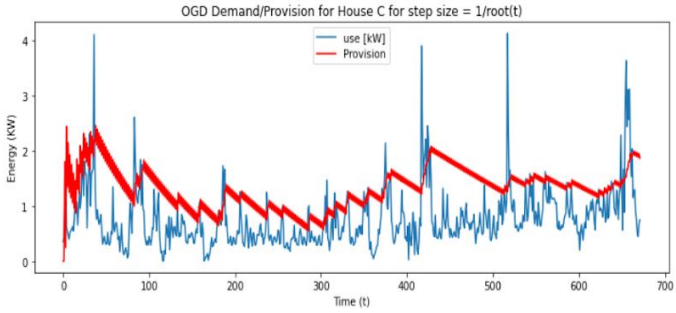


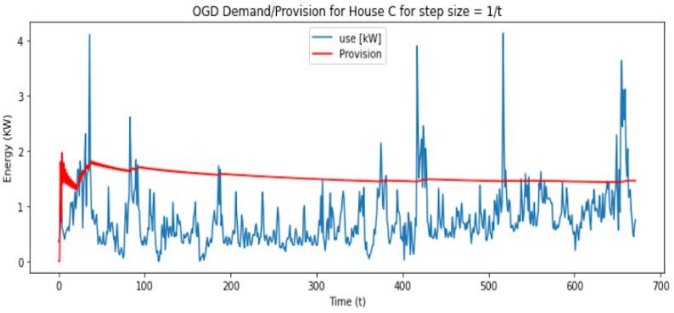

OGD Demand/Provision for House B for step size = 1/root(t)
use [kW]
Provision
Energy (KW)
Time (t)

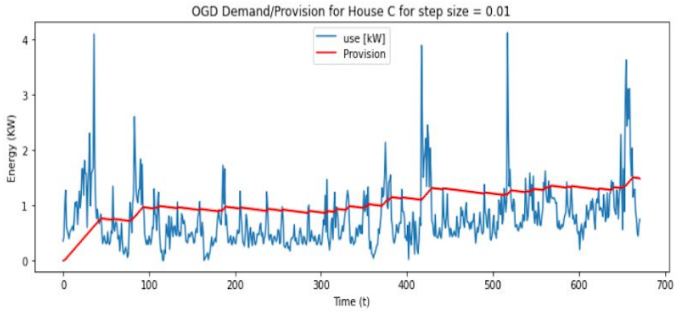

OGD Demand/Provision for House B for step size = 0.01
use [kW]
Provision
Energy (KW)
Time (t)

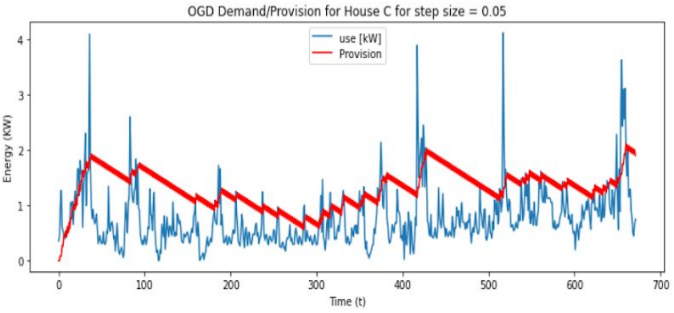

OGD Demand/Provision for House B for step size = 1/t
use [kW]
Provision
Energy (KW)
Time (t)

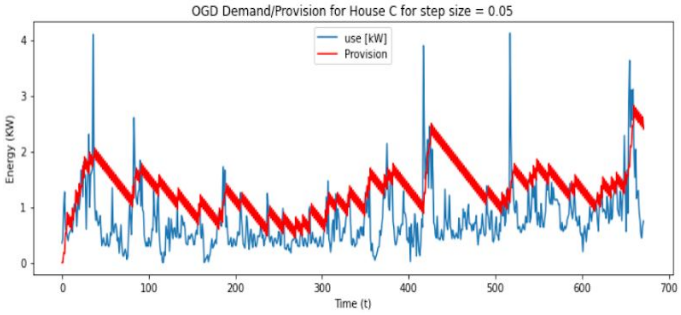

OGD Demand/Provision for House B for step size = 0.05
use [kW]
Provision
Energy (KW)
Time (t)

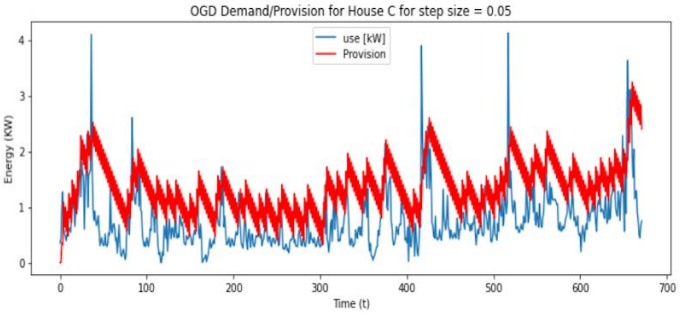

OGD Demand/Provision for House C for step size = 1/root(t)
use [kW]
Provision
Energy (KW)
Time (t)

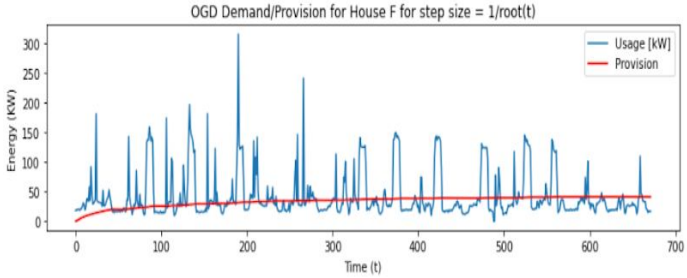

OGD Demand/Provision for House C for step size = 1/t
use [kW]
Provision
Energy (KW)
Time (t)

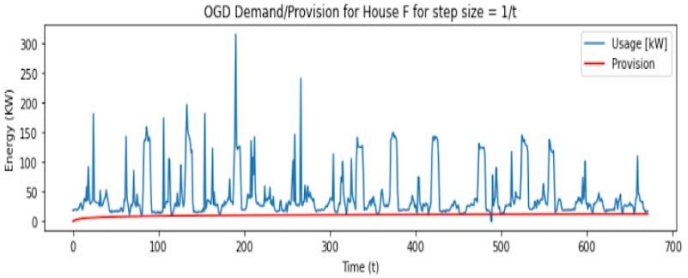

OGD Demand/Provision for House C for step size = 0.01
use [kW]
Provision
Energy (KW)
Time (t)

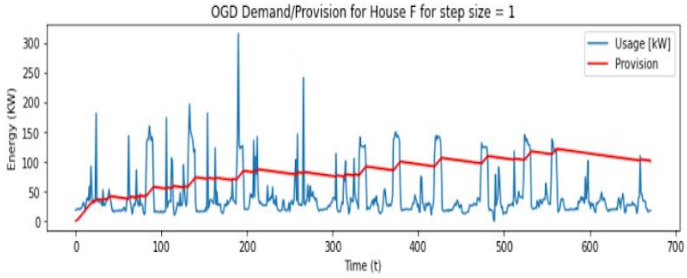

OGD Demand/Provision for House C for step size = 0.05
use [kW]
Provision
Energy (KW)
Time (t)

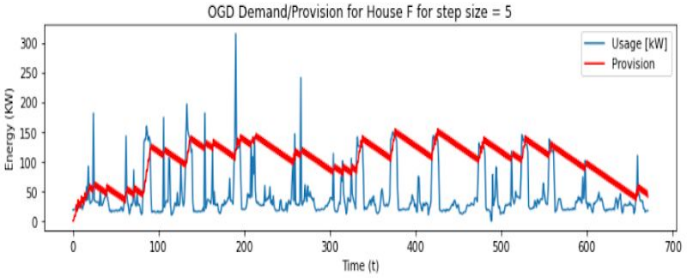

OGD Demand/Provision for House C for step size = 0.05
use [kW]
Provision
Energy (KW)
Time (t)


OGD Demand/Provision for House C for step size = 0.05
use [kW]
Provision
Energy (KW)
Time (t)


OGD Demand/Provision for House F for step size = 1/root(t)
Usage [kW]
Provision
Energy (KW)
Time (t)


OGD Demand/Provision for House F for step size = 1/t
Usage [kW]
Provision
Energy (KW)
Time (t)


OGD Demand/Provision for House F for step size = 1
Usage [kW]
Provision
Energy (KW)
Time (t)


OGD Demand/Provision for House F for step size = 5
Usage [kW]
Provision
Energy (KW)
Time (t)

2.2. *Receding Horizon Control (RHC):*

In this algorithm, y(t) is calculated using prediction results from two prediction algorithms used in the previous project. The prediction algorithms used are linear regression and random forests. Prediction values have been exported as csv files for ease of usage.

The objective function used in this algorithm is:

```
Localcost += ( price*x_window[i] + a*maximum( 0 , y_window[i]-x_window[i] ) +
b*abs(x_window[i] - x_window[i-1] ) )
```

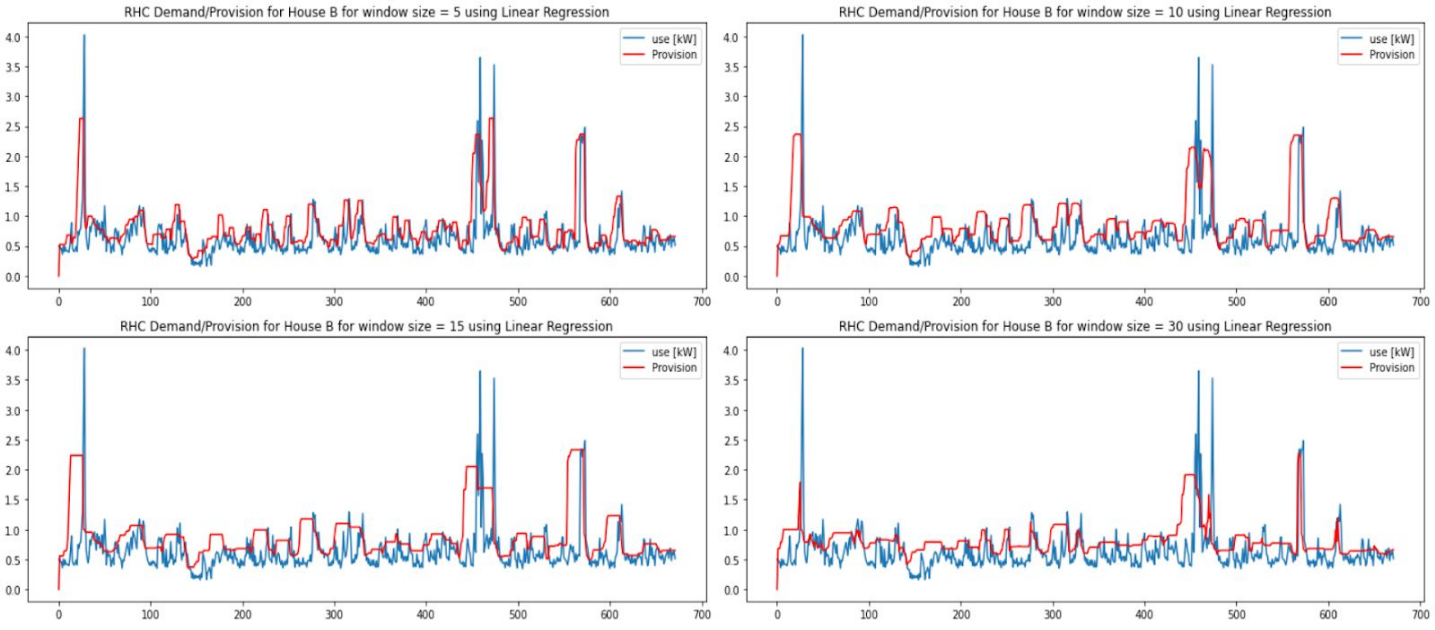
Here, x_window and y_window are the value of x and y for a given window size. We are minimizing the objective function using cvx. The corresponding value of x is then used to calculate the optimal cost. We have tried running the algorithm for a range of window sizes with values 5, 10, 15 and 30.

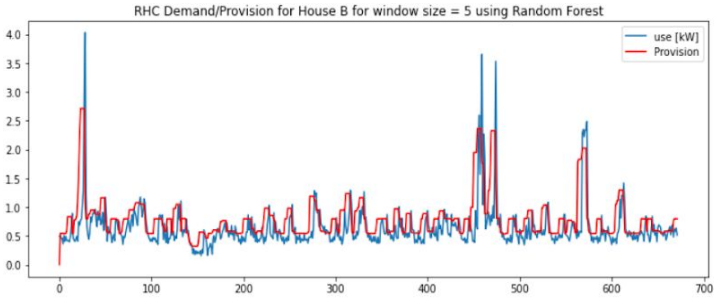
We get the following values of optimal cost using Receding Horizon Control algorithm:

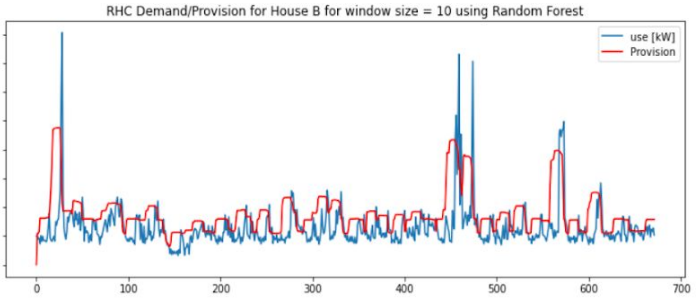
House	Window Size: Linear Regression				Window Size: Random Forest			
	5	10	15	30	5	10	15	30
B	244.244	231.201	228.668	226.033	247.687	235.004	236.028	229.923
C	341.721	324.507	313.675	305.067	346.508	318.538	298.995	297.391
F	23401.5	23179.5	22927.6	21999.4	22670.9	23339.2	22980.5	21493.7

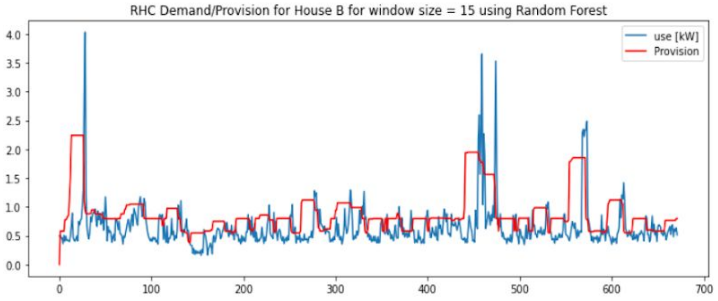
For all the three users, we get the optimal solution for window size = 30. For user B, we get the best solution using linear regression predictions, whereas for the other two users we get the best results using random forest predictions.

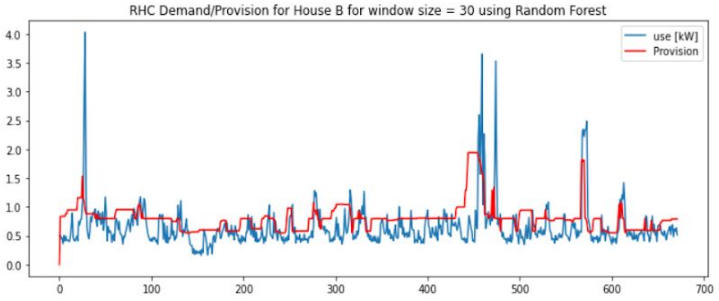
The demand/provision for the three users using Receding horizon control are as follows:

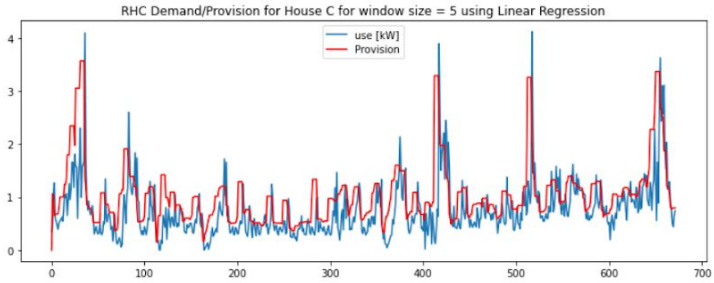


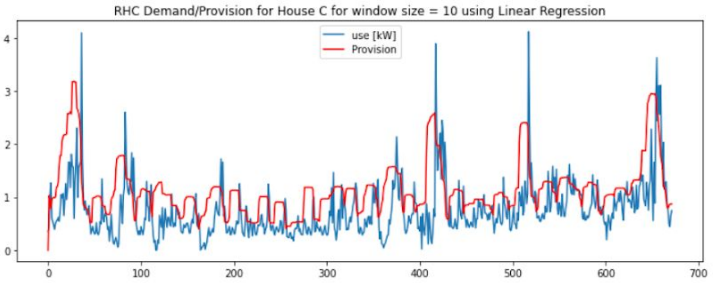

RHC Demand/Provision for House B for window size = 5 using Random Forest
use [kW]
Provision

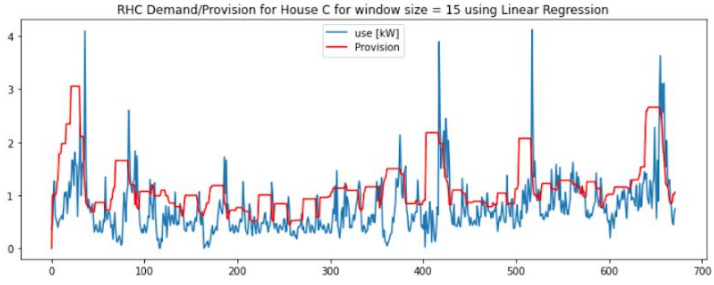

RHC Demand/Provision for House B for window size = 10 using Random Forest
use [kW]
Provision

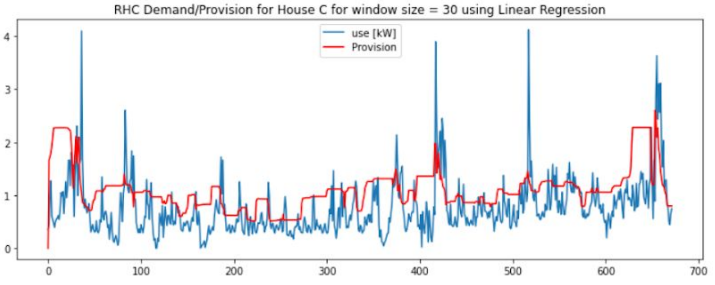

RHC Demand/Provision for House B for window size = 15 using Random Forest
use [kW]
Provision

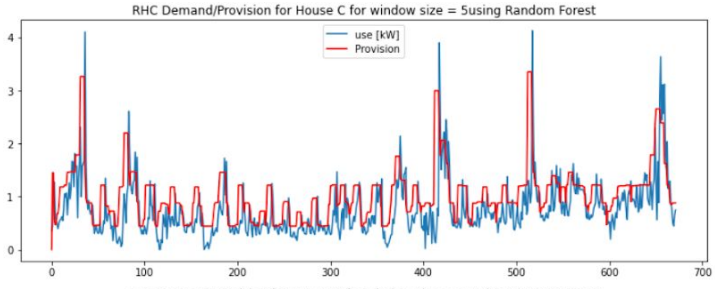

RHC Demand/Provision for House B for window size = 30 using Random Forest
use [kW]
Provision

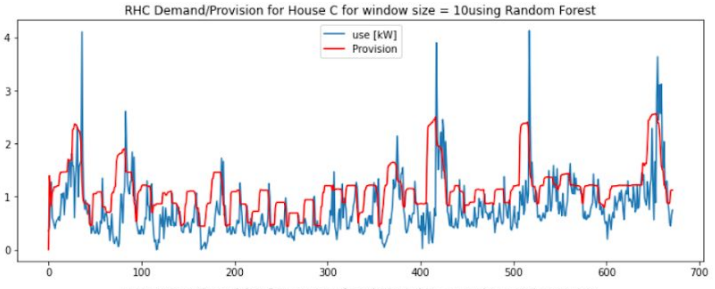

RHC Demand/Provision for House C for window size = 5 using Linear Regression
use [kW]
Provision

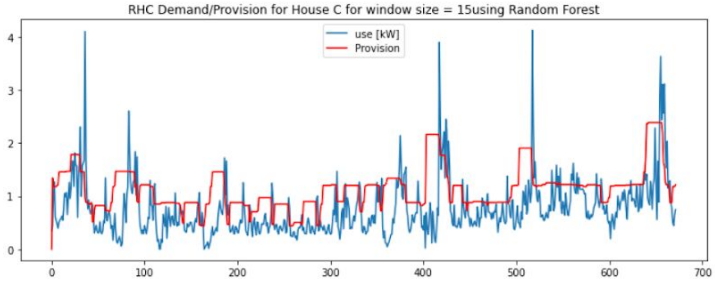

RHC Demand/Provision for House C for window size = 10 using Linear Regression
use [kW]
Provision

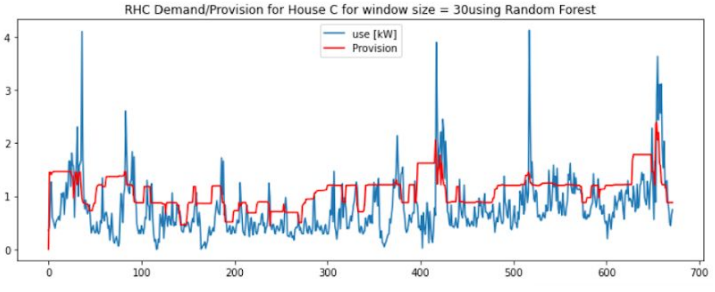

RHC Demand/Provision for House C for window size = 15 using Linear Regression
use [kW]
Provision

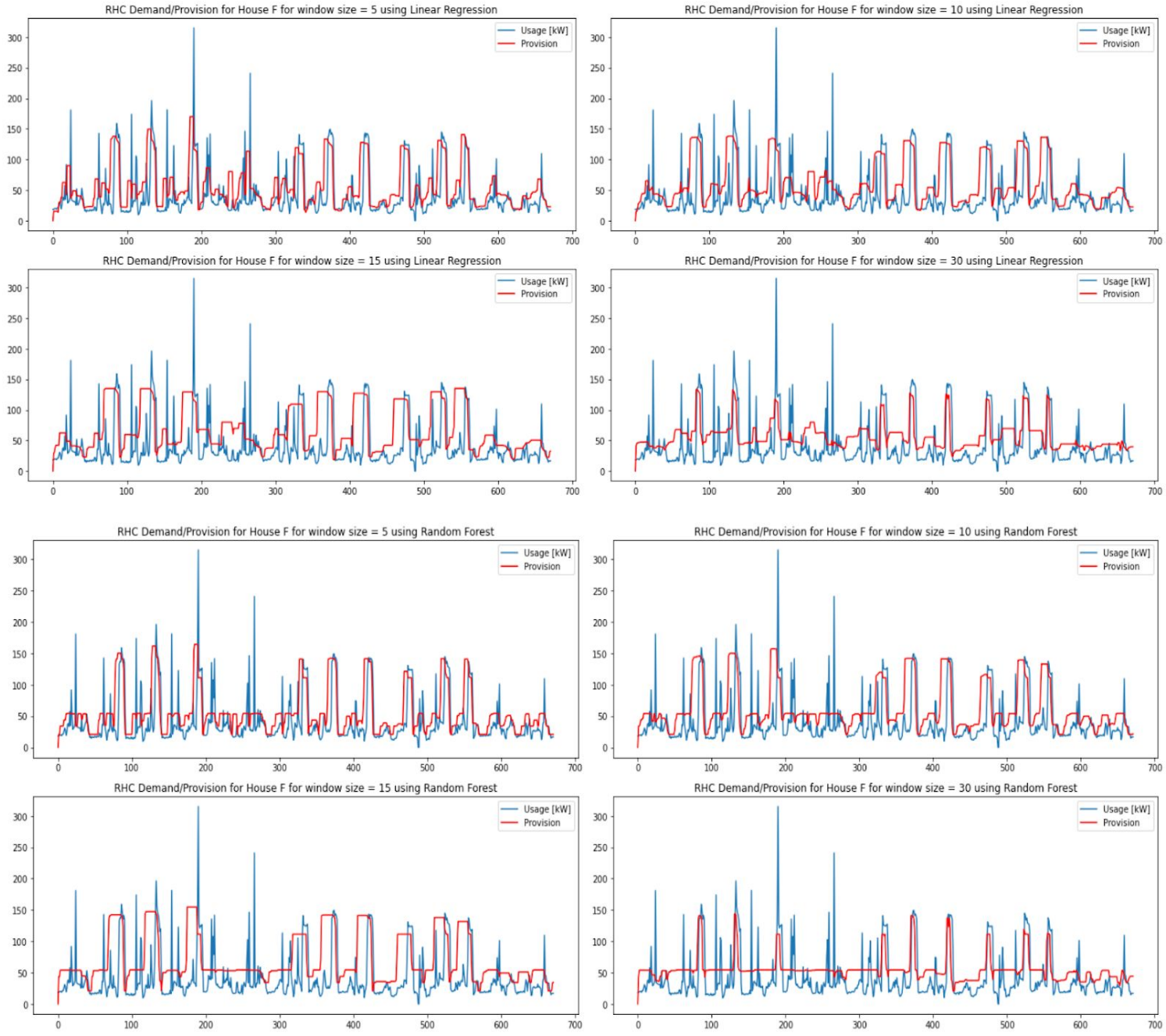

RHC Demand/Provision for House C for window size = 30 using Linear Regression
use [kW]
Provision


RHC Demand/Provision for House C for window size = 5using Random Forest
use [kW]
Provision


RHC Demand/Provision for House C for window size = 10using Random Forest
use [kW]
Provision


RHC Demand/Provision for House C for window size = 15using Random Forest
use [kW]
Provision


RHC Demand/Provision for House C for window size = 30using Random Forest
use [kW]
Provision



2.3. *Commitment Horizon Control (CHC):*

In this algorithm, y(t) is calculated using prediction results from two prediction algorithms used in the previous project. The prediction algorithms used are linear regression and random forests. Prediction values have been exported as csv files for ease of usage.

The objective function used in this algorithm is:

```
Localcost += ( price*x_window[i] + a*maximum( 0 , y_window[i]-x_window[i] ) +
b*abs(x_window[i] - x_window[i-1] ) )
```

This algorithm builds up on RHC. Here, x_window and y_window are the value of x and y for a given window size. We are minimizing the objective function using cvx. The corresponding value of x is then used to calculate the optimal cost. We are setting window size as 30, since

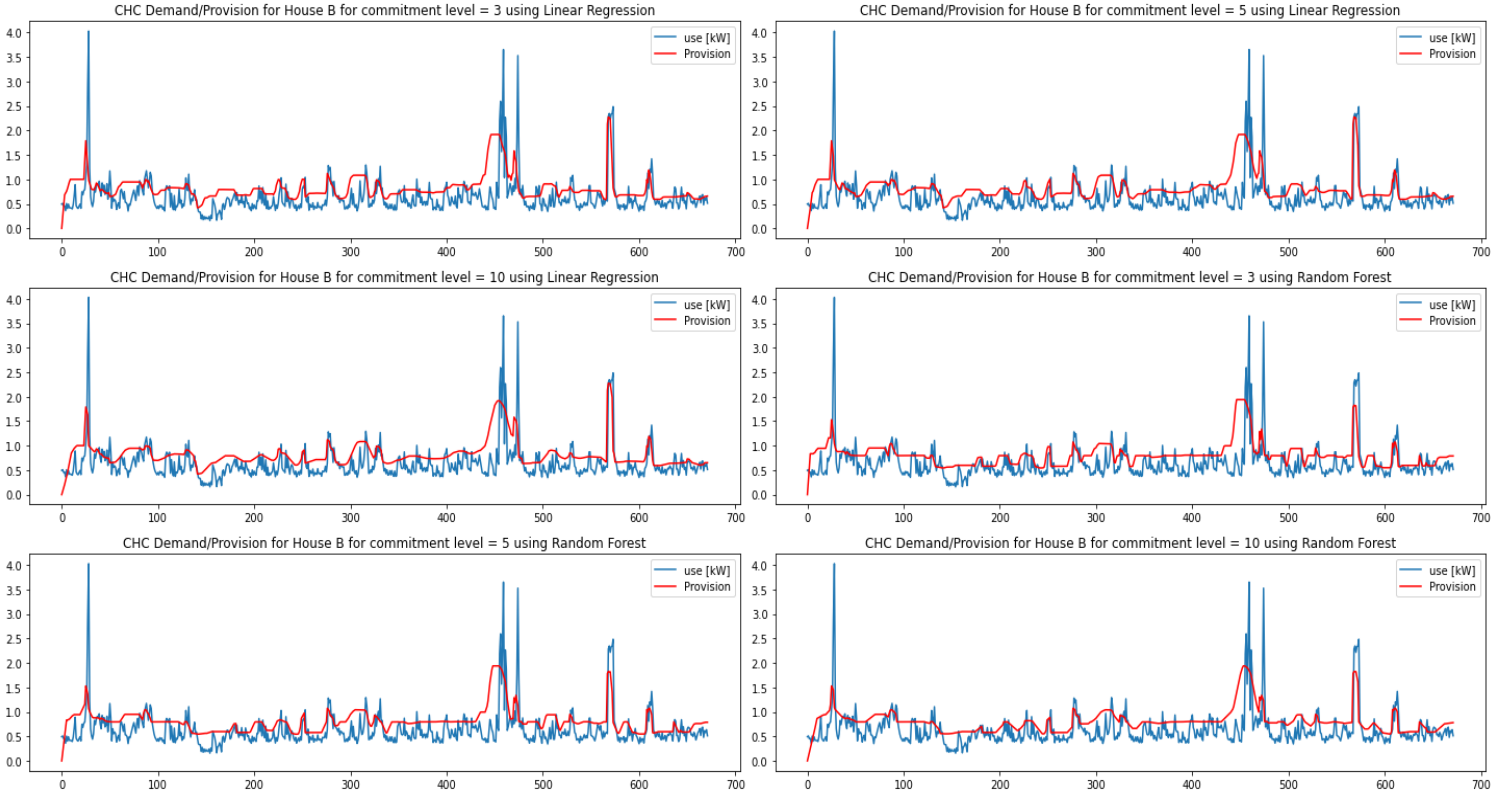
we get optimal cost for this window size using RHC. We are running this algorithm over a range of commitment horizon values of 3, 5 and 10. We then find the optimal solution for the commitment horizon with least value of objective function.

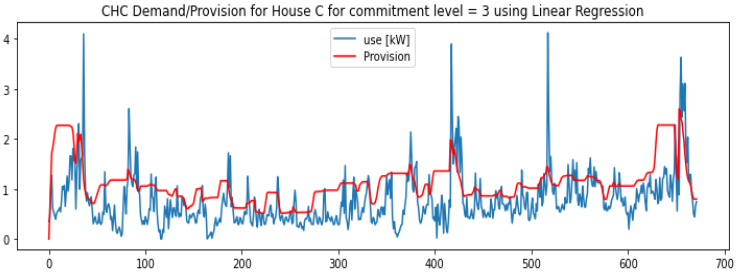
We get the following values of optimal cost using Commitment Horizon Control algorithm:

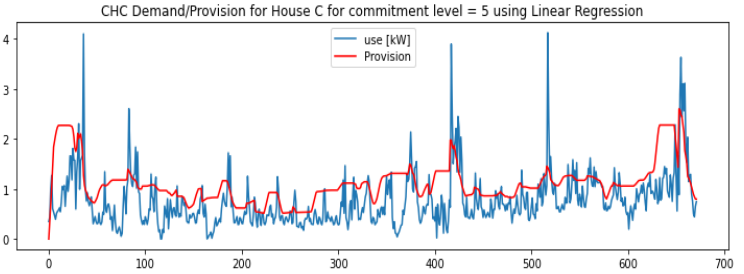
House	Commitment Level: Linear Regression			Commitment Level: Random Forest		
	3	5	10	3	5	10
B	210.725	204.963	198.003	210.516	202.996	194.911
C	279.548	269.875	258.784	273.581	265.806	256.300
F	20387	19709	18999	20011	19393	18831

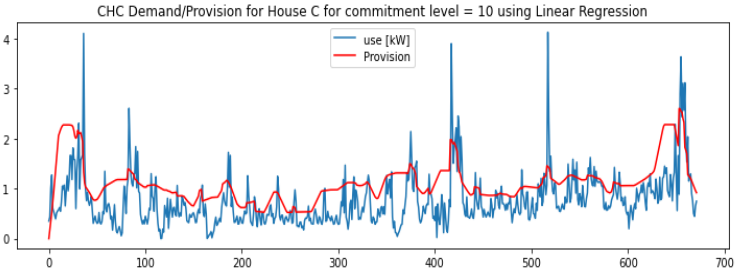
For all the three users, we get the optimal solution for commitment level = 10 using random forest predictions.

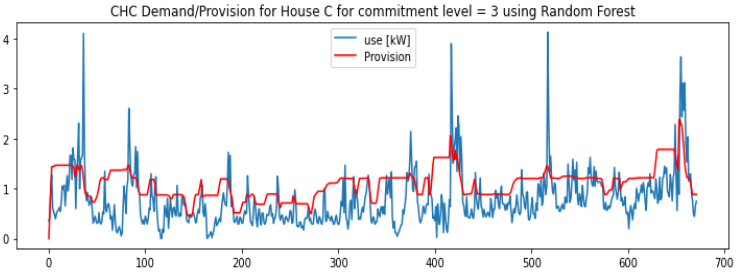
The demand/provision for the three users using Commitment horizon control are as follows:

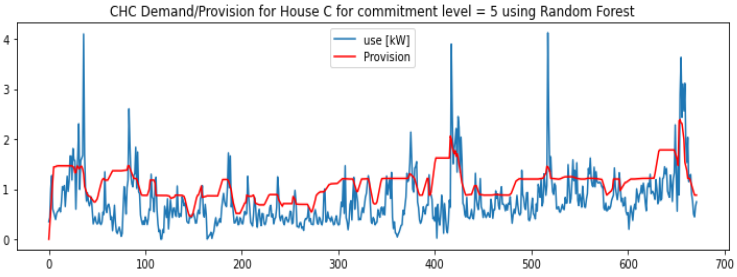


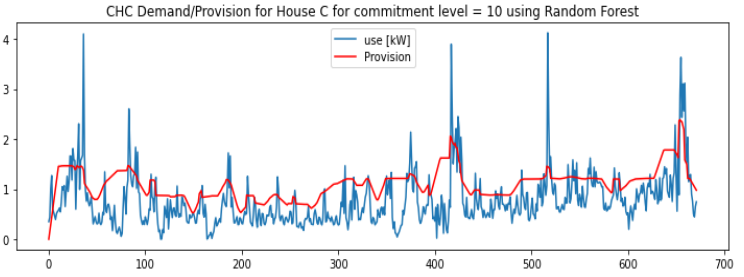

CHC Demand/Provision for House C for commitment level = 3 using Linear Regression
use [kW]
Provision

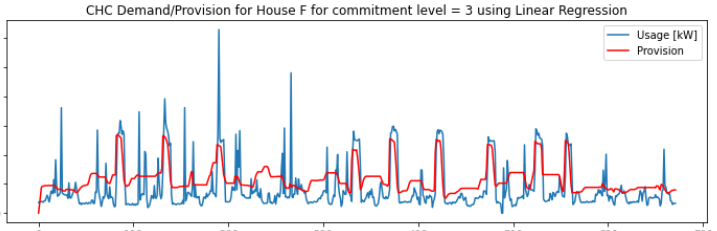

CHC Demand/Provision for House C for commitment level = 5 using Linear Regression
use [kW]
Provision

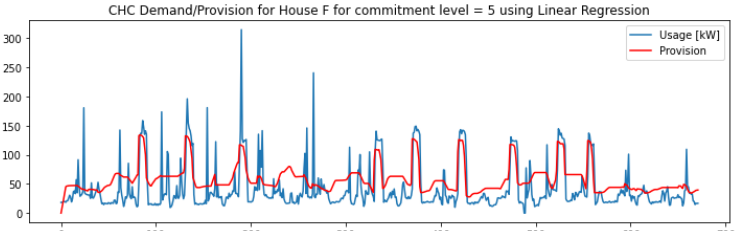

CHC Demand/Provision for House C for commitment level = 10 using Linear Regression
use [kW]
Provision

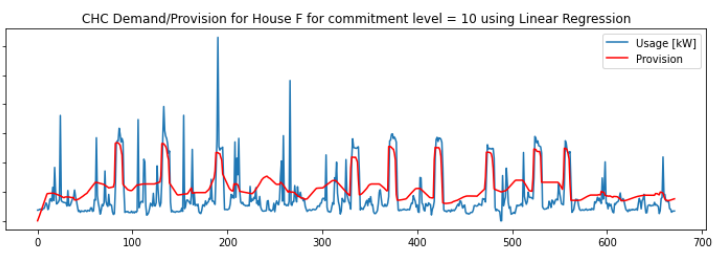

CHC Demand/Provision for House C for commitment level = 3 using Random Forest
use [kW]
Provision


CHC Demand/Provision for House C for commitment level = 5 using Random Forest
use [kW]
Provision

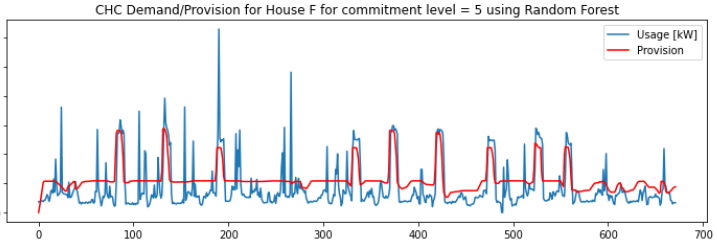

CHC Demand/Provision for House C for commitment level = 10 using Random Forest
use [kW]
Provision

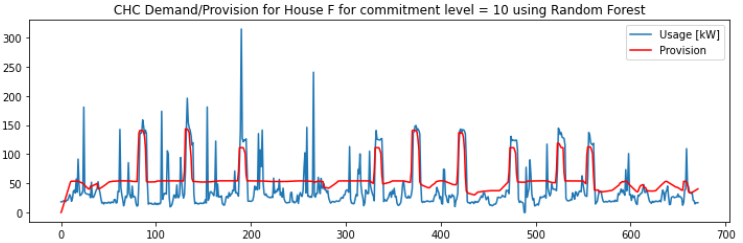

CHC Demand/Provision for House F for commitment level = 3 using Linear Regression
Usage [kW]
Provision


CHC Demand/Provision for House F for commitment level = 5 using Linear Regression
Usage [kW]
Provision


CHC Demand/Provision for House F for commitment level = 10 using Linear Regression
Usage [kW]
Provision


CHC Demand/Provision for House F for commitment level = 3 using Random Forest
Usage [kW]
Provision


CHC Demand/Provision for House F for commitment level = 5 using Random Forest
Usage [kW]
Provision


CHC Demand/Provision for House F for commitment level = 10 using Random Forest
Usage [kW]
Provision

3. Comparison of Algorithm Costs:

We are comparing costs of our algorithms to those of offline static and dynamic solutions. Comparisons are made in terms of optimal costs as well as static and dynamic regret values.

A summary of optimal costs for all the algorithms is as follows:

House	Offline Static	Offline Dynamic	Online Gradient Descent	Receding Horizon Control	Commitment Horizon Control
B	193.024	160.471	223.688	226.033	194.911
C	257.280	209.799	293.986	297.391	256.300
F	19154.158	14744.553	23832	21493.7	18831

We can see that for all three houses we have the lowest cost for offline dynamic solution. This proves that offline dynamic solution can be considered a lower bound for the objective function.

Among all the online algorithms (OGD, RHC and CHC), we can see that CHC has the lowest cost associated followed by RHC followed by OGD. In some cases, online algorithms performed better than the corresponding offline static solution (CHC < Offline static for house C and F).

The same can be realised by finding static and dynamic regret, which are computed as follows:
Static Regret = Cost of Online Algorithm - Cost of Static Offline solution
Dynamic Regret = Cost of Online Algorithm - Cost of Dynamic Offline solution

The static and dynamic regrets for our users are as follows:

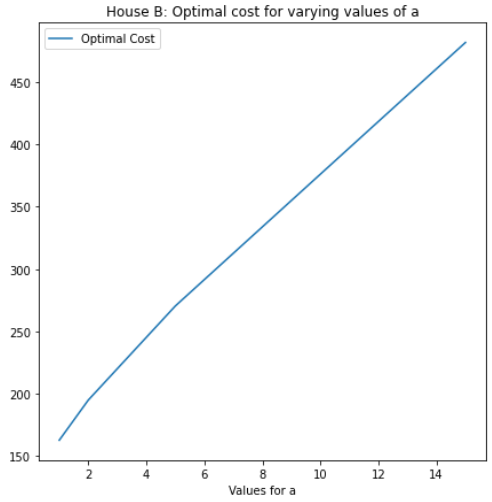
House	OGD Static Regret	OGD Dynamic Regret	RHC Static Regret	RHC Dynamic Regret	CHC Static Regret	CHC Dynamic Regret
B	30.664192	63.217623	33.009167	65.562598	1.886614	34.440045
C	36.706255	84.187006	40.110686	87.591438	-0.980290	46.500461
F	4678.523396	9088.128676	2339.587068	6749.192348	-322.983100	4086.622180

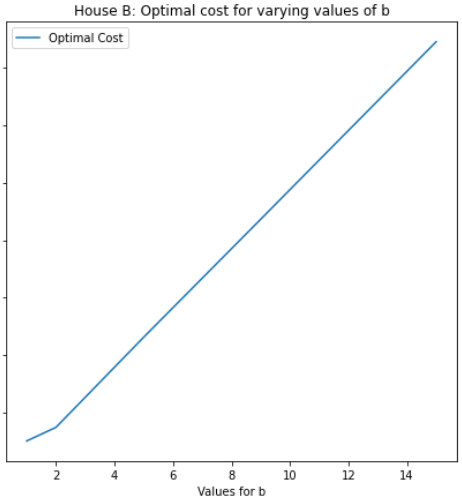
4. Varying values of ‘a’ and ‘b’:

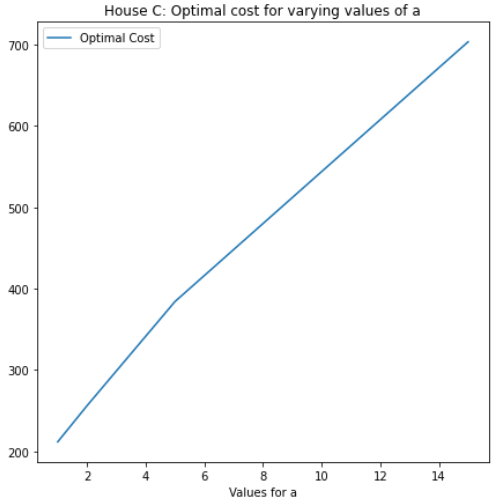
We tried multiple values of a (penalty) and b (switching cost) to observe their effect on the overall objective function and algorithm performance. To do so, we kept either of the two variables constant and changed the other variable over a range of values of 1, 2, 5, 9, 12, 15.

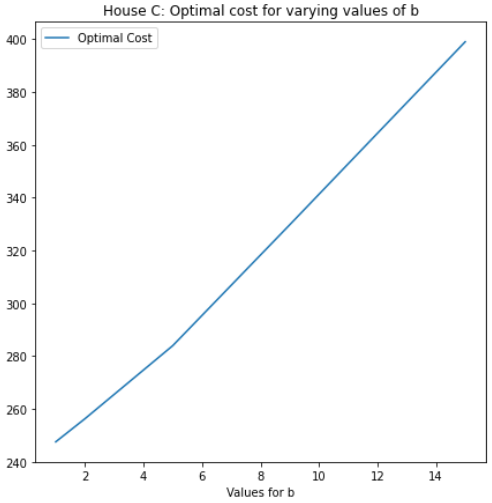
For both the variables, we found that the cost increases with increase in values of a and b respectively.

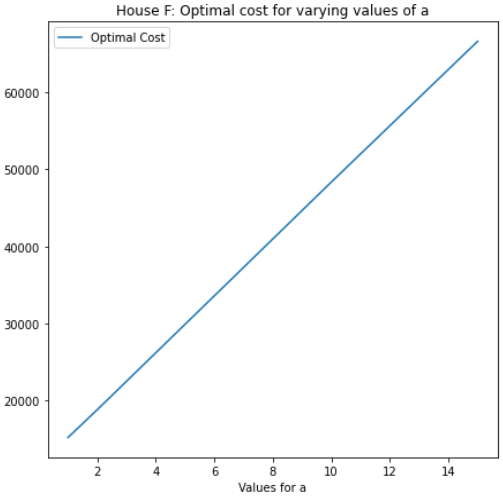
We can see corresponding visualizations for change in a and b and their effect on cost as:

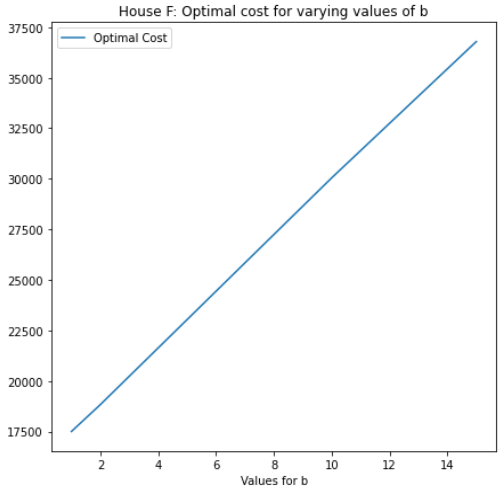












5. Algorithm Selection:

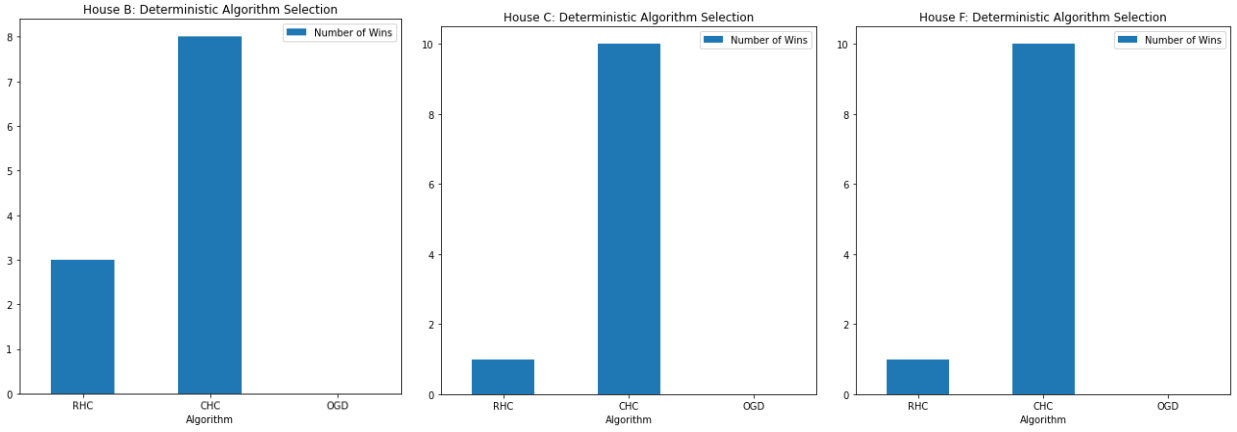
We have come up with a deterministic and a randomized algorithm to choose between our existing online algorithms in order to choose the most efficient algorithm.

5.1 Deterministic Algorithm Selection:

The algorithm works as follows:

1. Set delta to a particular fixed value, global optimal cost to zero.
2. Set a fixed optimal window size and commitment horizon.
3. Iterate over all values of time in steps of delta.
4. At each iteration, find the optimal cost for that chunk of data using RHC, CHC and OGD.
5. Find the algorithm with the lowest cost, and declare it the winner for that iteration and assign local optimal cost as it's cost for the iteration.
6. Increment the global optimal cost by the local optimal cost for that iteration.

In the end, we get a total count of the number of times an algorithm has won. We can see the algorithm's performance as follows:



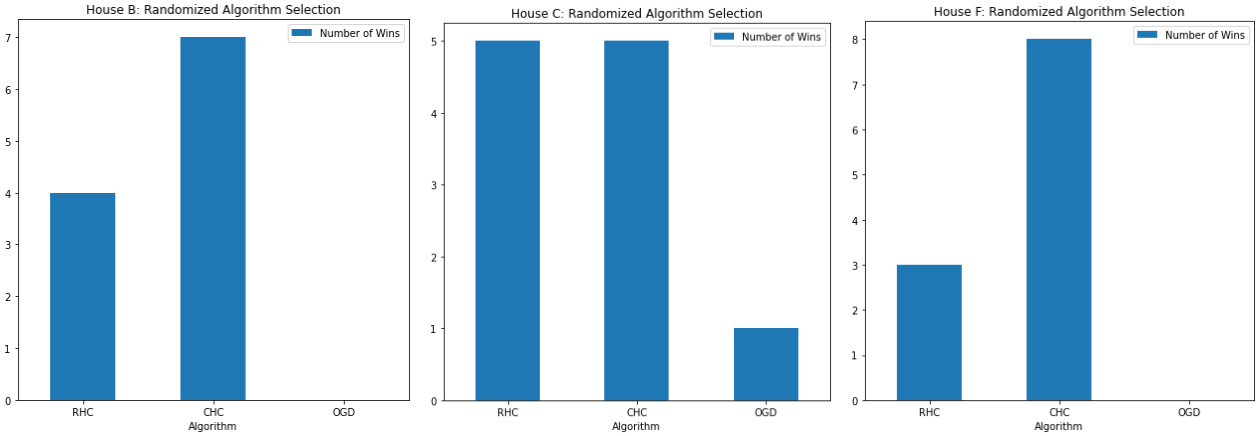
We can see that CHC was optimal the most number of times followed by RHC.

5.2 Randomized Algorithm Selection:

The algorithm works as follows:

1. Set delta to a particular fixed value, global optimal cost to zero.
2. Set a fixed optimal window size and commitment horizon.
3. Iterate over all values of time in steps of delta.
4. At each iteration, find the optimal cost for that chunk of data using RHC, CHC and OGD.
5. Now depending on the cost of an algorithm, assign it's probability to a particular value (0.6 for lowest, 0.3 for second lowest, 0.1 for highest).
6. Choose an algorithm as a winner for the iteration based on their probability distribution and assign local optimal cost as it's cost for the iteration.
7. Increment the global optimal cost by the local optimal cost for that iteration.

In the end, we get a total count of the number of times an algorithm has won. We can see the algorithm's performance as follows:



Even in this case, we can see that CHC was optimal the most number of times followed by RHC.

**Please refer to the ipynb file for detailed insights and code.*