Fahad Fiaz – (303141) – G2

System Info:

Processor	i7-5500U , 2.40GHz
Cores	4
Operating system	Windows 64 Bit
Ram	8GB
Programming Language	Python 3.7.7

Exercise 1: Parallel Linear Regression?

Steps:

1. Consider 1st process as master node and other as slave node.

2. Master process will first read data from files. Since there are multiple files in dataset so we use glob library to read names of multiple files at once. Then the dataset is randomly shuffled and preprocessed to get X as Feature matrix and Y as Prediction Vector. Then we split data into 70% train and 30% test data. Then master process divides the training data into equal chunks according to number of workers and scatters it to child Nodes.

```python
# Pre_processing function to convert data to specific format
def pre_processing(feature_example):
    total_features = np.zeros(482)
    y = 0.0
    processed_features = feature_example.split(" ")
    if (processed_features[0].strip() != ''):
        y = float(processed_features[0])
    for i in range(len(processed_features)):
        split_features = processed_features[i].split(":")
        if (len(split_features) == 2):
            total_features[int(split_features[0])] = int(split_features[1])
    return total_features, y
```

```python
Total_data = []
    path = "dataset"
    All_Files = glob.glob(path + "/*.txt")
    for name in All_Files:
        File = open(name)
        File_data = File.read().split("\n")
        Total_data.append(File_data)
    np.random.shuffle(Total_data)  # shuffle complete dataset randomly
    for i in range(len(Total_data)):
        for j in range(len(Total_data[i])):
            x, y = pre_processing(Total_data[i][j])
            X.append(x)
            Y.append(y)
    X = np.array(X)  # feature matrix
    Y = np.array(Y)  # prediction vector
    FullDataset = X.shape[0]

    # splitting Dataset in Test/Train format
    X_Train = X[:(FullDataset * 70) // 100, :]
    X_Test = X[(FullDataset * 70) // 100:, :]
    Y_Train = Y[:(FullDataset * 70) // 100]
    Y_Test = Y[(FullDataset * 70) // 100:]

    # partitioning dataset
    data_size = len(X_Train)
    partition = data_size // size
    for process in range(0, size):  # partition data in equal portions
        start = process * partition
        end = (process + 1) * partition
        if process == size - 1 and end != data_size:
            end = data_size
        X_Chunk.append(X_Train[start:end])
        Y_Chunk.append(Y_Train[start:end])
```

3. Slave processes received chunk of Feature Matrix and Prediction Vector. Then they use SGD algorithm to find parameters (**Beta**) of model.

```
# scattering data to child processes
Feature_chunk = np.array(comm.scatter(X_Chunk))
Prediction_chunk = np.array(comm.scatter(Y_Chunk))

Parameters = SGD(Feature_chunk, Prediction_chunk, alpha)
```

```
def SGD(X_Train, Y_Train, alpha):  # function to apply Stochastic Gradient Descent algorithm
    if flag == False:  # checking if you are running sgd for 1st epoch else Parameters will get value from broadcasting
        Parameters = np.zeros(len(X_Train[0]))
    for i in range(len(X_Train)):
        my_prediction = np.dot(Parameters.T, X_Train[i])
        error = Y_Train[i] - my_prediction
        Parameters = Parameters + (2 * (alpha * error))
    return Parameters
```

4. Master process then gathers the local parameters (**Beta**) of model from slave workers and averages them and broadcast the new global model parameters (**Beta**) to each worker for the next epoch. We also calculate RMSE loss on Test data using averaged model parameters.

```
# Master node gathering parameters from child processes
All_Local_Parameters = comm.gather(Parameters, root=Master)

if (rank == Master):
    if (size > 1):  # checking if there were multiple child processes
        Global_Parameters_Mean = np.mean(All_Local_Parameters,
                                         axis=0)  # Master node calculate Global mean fo parameters
        Loss.append(RMSE(X_Test, Y_Test, Global_Parameters_Mean))  # calculating Loss
    else:
        Global_Parameters_Mean = All_Local_Parameters  # If there was single process
else:
    Global_Parameters_Mean = None

Global_Parameters_Mean = comm.bcast(Global_Parameters_Mean, root=Master) # sending average of parameters to child proceses for next epoch

if len(Global_Parameters_Mean) > 0:
    flag = True
    Parameters = Global_Parameters_Mean
Epoch = Epoch + 1
def RMSE(X, Y, Global_Parameters_Mean):  # function to calculate RMSE
    Predictions = []
```

```python
for i in range(len(X)):
    Predictions.append(np.dot(X[i], Global_Parameters_Mean.T))
Final_Predictions = np.reshape(np.array(Predictions), -1)
Error = sum((Y - Final_Predictions) ** 2)
return Error
```

Code working for arbitrary number of clusters and doing parallel processing:

In [2]:
```
!mpiexec -n 1 python exercise4.py
```
```
Train/Test Loss:  11693.26126902823
Total working time: 31.00262250006199
```

In [3]:
```
!mpiexec -n 2 python exercise4.py
```
```
Train/Test Loss:  12091.201923924275
Total working time: 25.588684600079432
```

In [4]:
```
!mpiexec -n 3 python exercise4.py
```
```
Train/Test Loss:  12488.4175282204
Total working time: 18.973370800027624
```

In [5]:
```
!mpiexec -n 4 python exercise4.py
```
```
Train/Test Loss:  12762.193729963828
Total working time: 20.494305700063705
```

In [6]:
```
!mpiexec -n 5 python exercise4.py
```

```
Train/Test Loss:  12914.32067656716
Total working time: 20.691015099873766
```

In [7]:
```
!mpiexec -n 6 python exercise4.py
```

```
Train/Test Loss:  12984.266807230139
Total working time: 19.21289510000497
```

In [8]:
```
!mpiexec -n 7 python exercise4.py
```

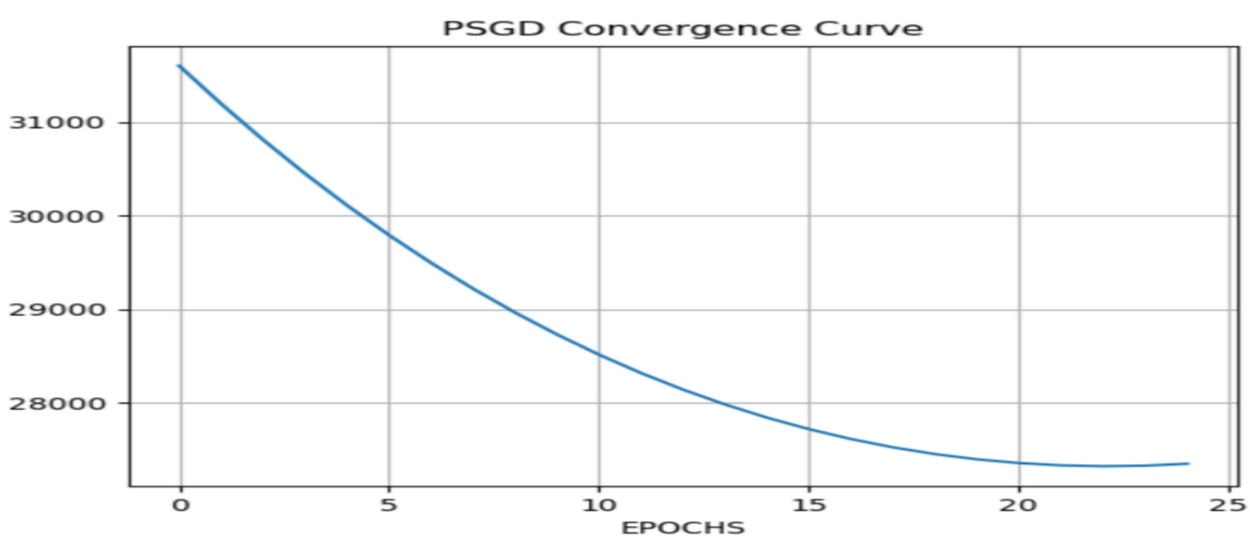
```
Train/Test Loss:  13143.402004668993
Total working time: 18.94527080003172
```

In [9]:
```
!mpiexec -n 8 python exercise4.py
```

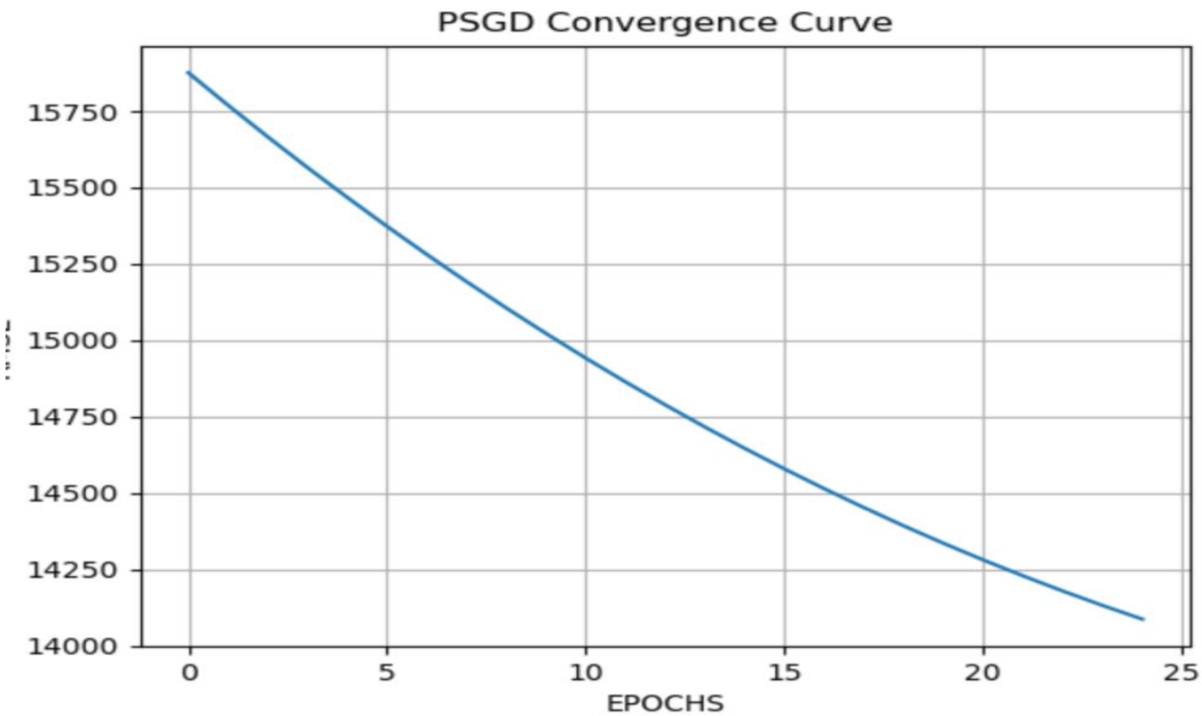
```
Train/Test Loss:  13147.863157772854
Total working time: 19.287448999937624
```

Convergence Graphs:

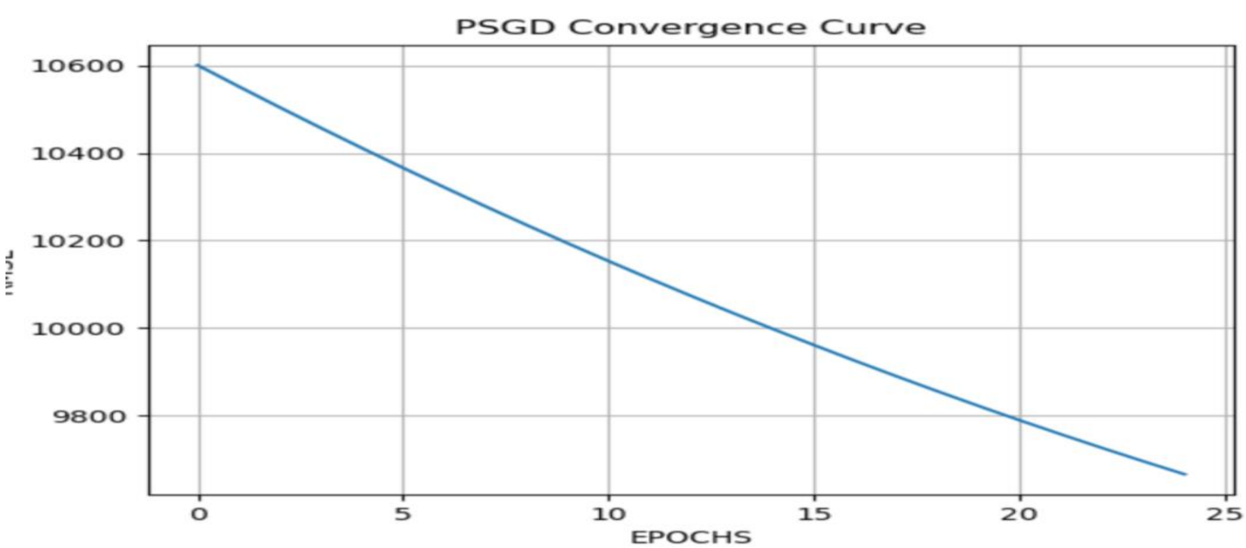
P=1



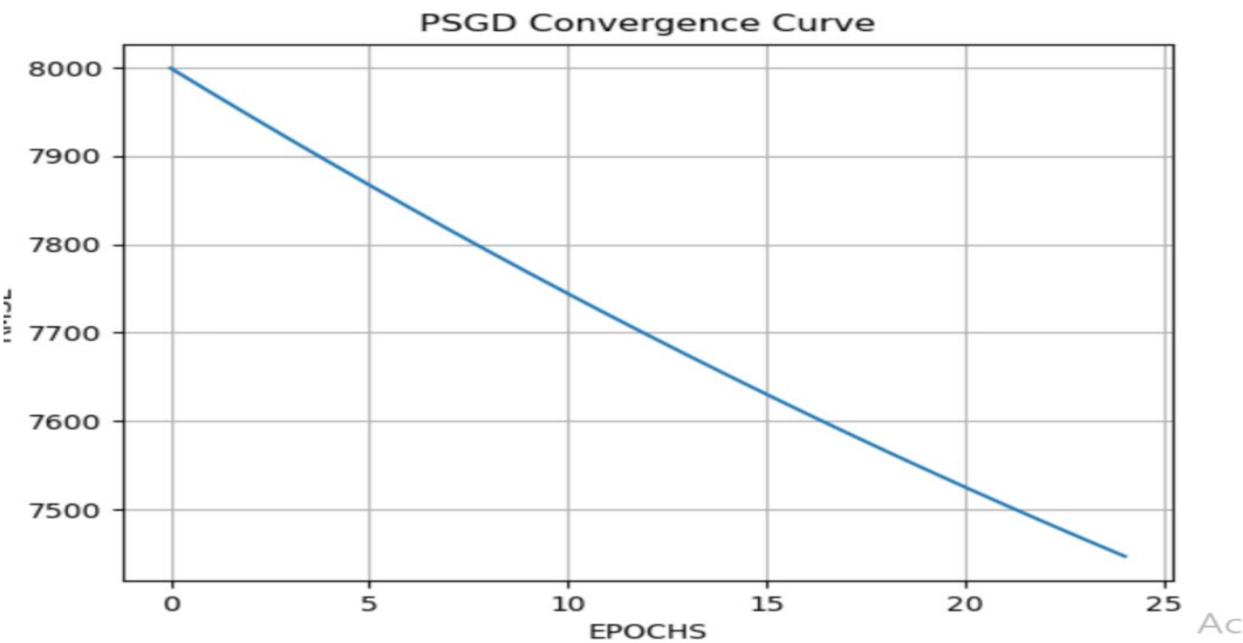
P=2


PSGD Convergence Curve
15750
15500
15250
15000
14750
14500
14250
14000
0
5
10
15
20
25
EPOCHS

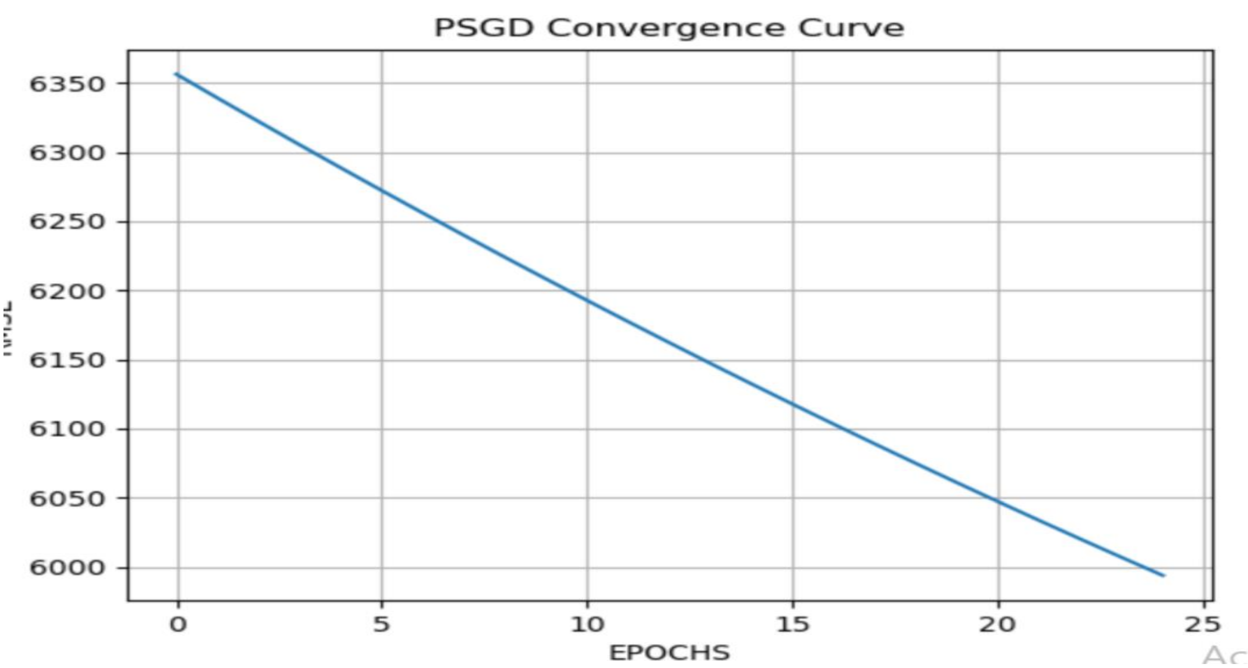
P=3


PSGD Convergence Curve
10600
10400
10200
10000
9800
0
5
10
15
20
25
EPOCHS

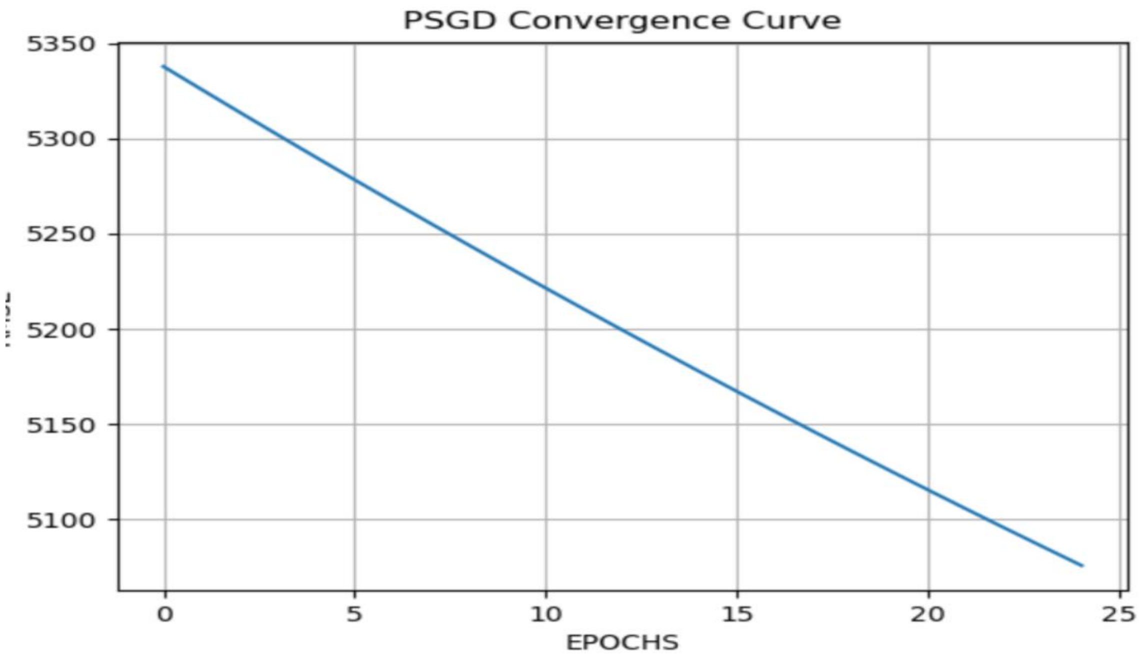
<u>P=4</u>


PSGD Convergence Curve
8000
7900
7800
7700
7600
7500
0
5
10
15
20
25
EPOCHS

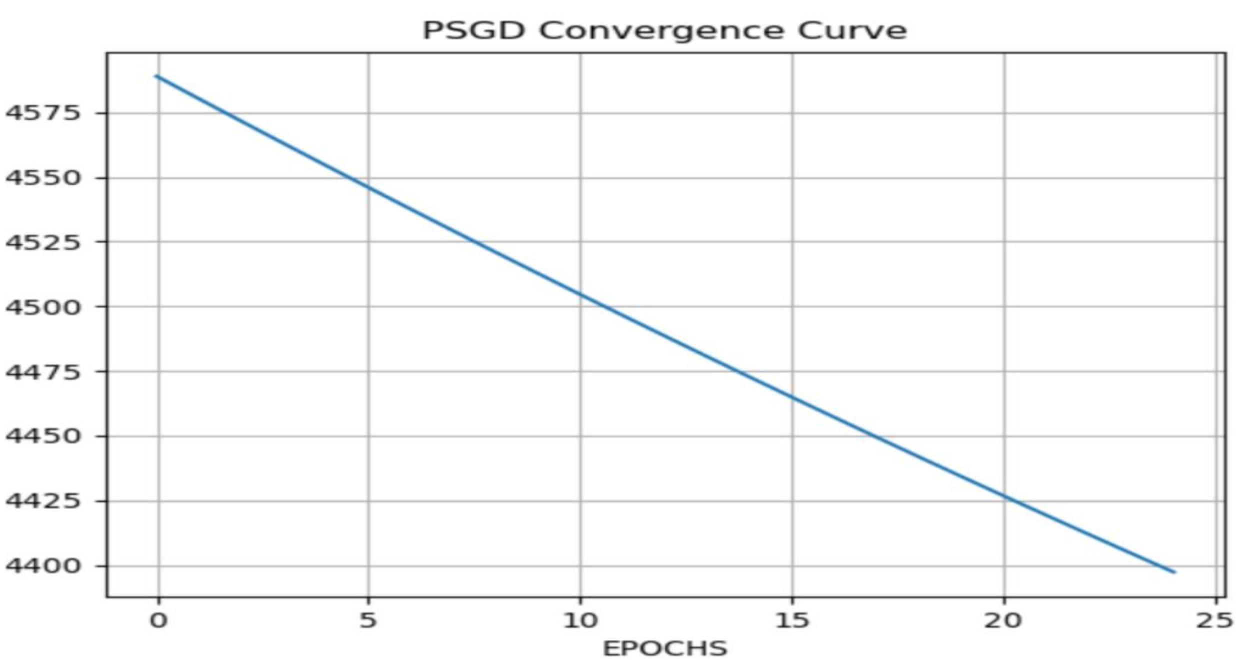
<u>P=5</u>


PSGD Convergence Curve
6350
6300
6250
6200
6150
6100
6050
6000
0
5
10
15
20
25
EPOCHS

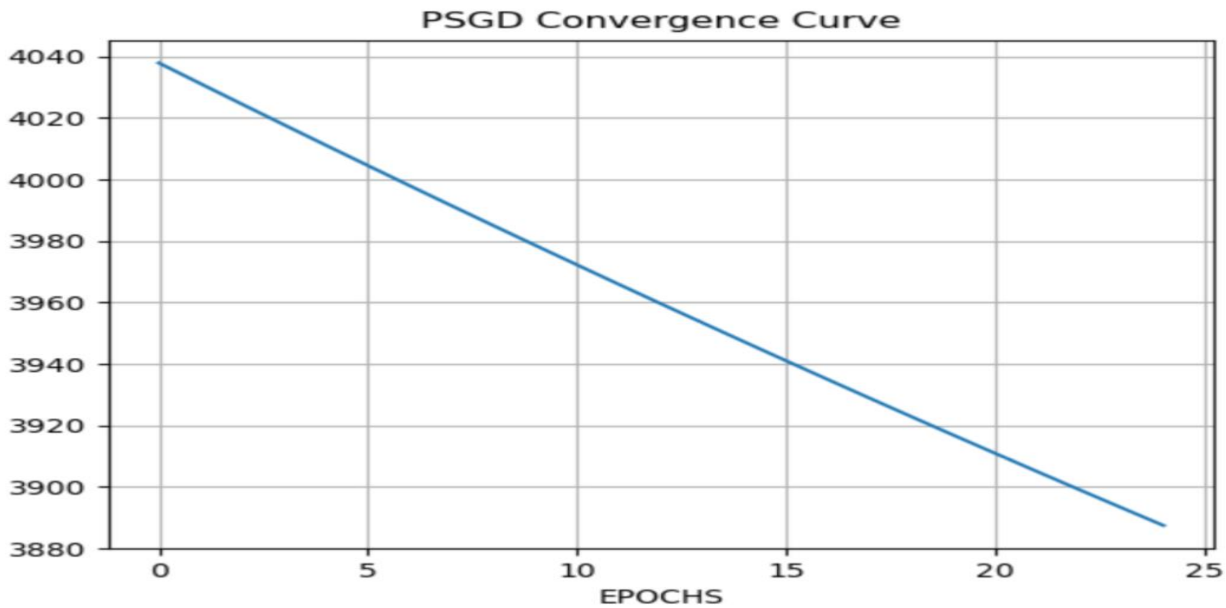
P=6


PSGD Convergence Curve
5350
5300
5250
5200
5150
5100
0
5
10
15
20
25
EPOCHS

P=7


PSGD Convergence Curve
4575
4550
4525
4500
4475
4450
4425
4400
0
5
10
15
20
25
EPOCHS

P=8


PSGD Convergence Curve
4040
4020
4000
3980
3960
3940
3920
3900
3880
0
5
10
15
20
25
EPOCHS

Performance analysis by epochs vs time graph:

P=1:

```
In [10]: !mpiexec -n 1 python exercise4.py
         Total working time: 12.544648499926552
         Epochs:   5
```

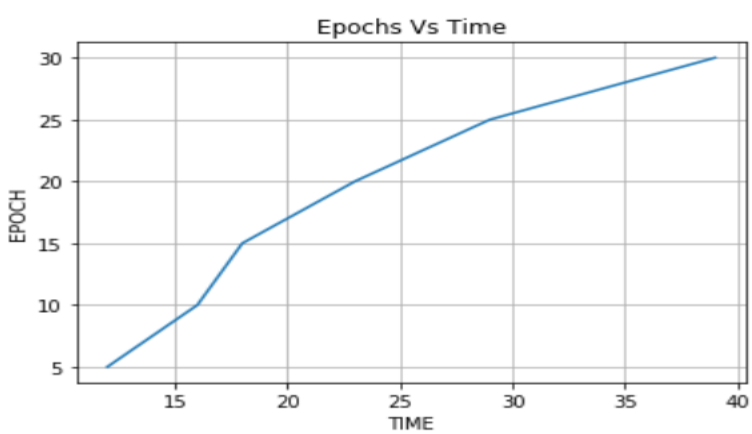
```
In [11]: !mpiexec -n 1 python exercise4.py
         Total working time: 16.875064900144935
         Epochs:   10
```

```
In [13]: !mpiexec -n 1 python exercise4.py
         Total working time: 18.514818799914792
         Epochs:   15
```

```
In [14]: !mpiexec -n 1 python exercise4.py
         Total working time: 23.04455990018323
         Epochs:   20
```

```
In [18]: !mpiexec -n 1 python exercise4.py
         Total working time: 29.152129899943247
         Epochs:   25
```

```
In [19]: !mpiexec -n 1 python exercise4.py
         Total working time: 39.83932360005565
         Epochs:   30
```



P=2:

In [26]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 12.863581700017676
Epochs:   5
```

In [29]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 15.80437029991299
Epochs:   10
```

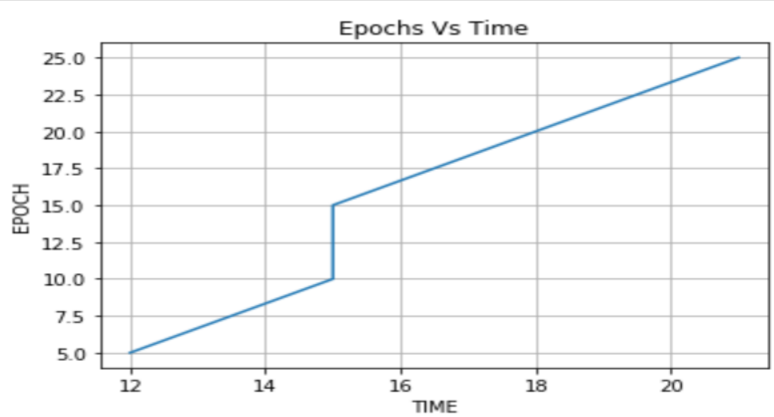
In [30]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 15.756372400093824
Epochs:   15
```

In [31]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 18.7912723000627
Epochs:   20
```

In [32]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 21.139919500099495
Epochs:   25
```

In [19]:
```
!mpiexec -n 2 python exercise4.py
```
```
Total working time: 39.83932360005565
Epochs:   30
```

In [34]:
```
from matplotlib import pyplot as plt
plt.title('Epochs Vs Time')
plt.xlabel('TIME')
plt.ylabel('EPOCH')
plt.grid()
plt.plot([12,15,15,18,21],[5,10,15,20,25])
plt.show()
```



P=3:

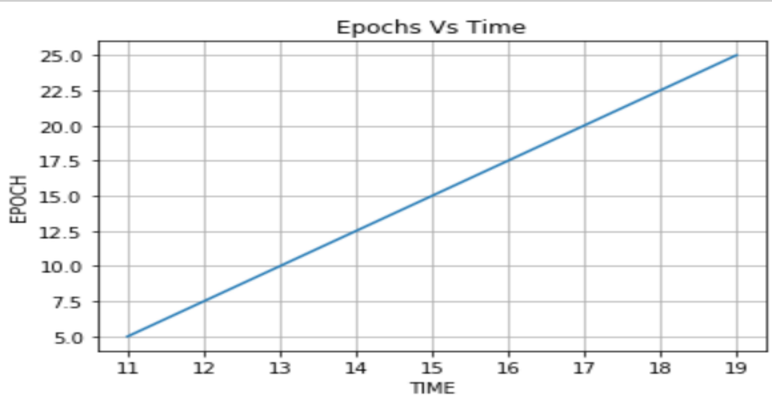
In [35]:
```
!mpiexec -n 3 python exercise4.py
```
```
Total working time: 11.939011399867013
Epochs:  5
```

In [36]:
```
!mpiexec -n 3 python exercise4.py
```
```
Total working time: 13.844968799967319
Epochs:  10
```

In [37]:
```
!mpiexec -n 3 python exercise4.py
```
```
Total working time: 15.370876600034535
Epochs:  15
```

In [38]:
```
!mpiexec -n 3 python exercise4.py
```
```
Total working time: 17.591000499902293
Epochs:  20
```

In [42]:
```
from matplotlib import pyplot as plt
plt.title('Epochs Vs Time')
plt.xlabel('TIME')
plt.ylabel('EPOCH')
plt.grid()
plt.plot([11,13,15,17,19],[5,10,15,20,25])
plt.show()
```



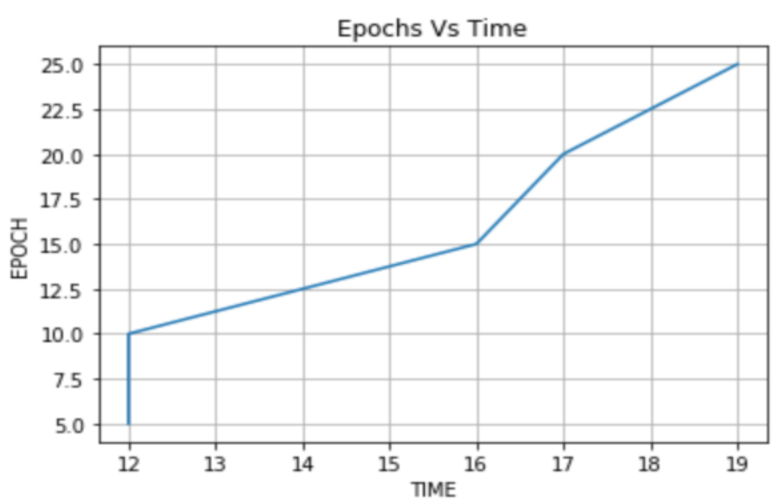
P=4:

```
In [52]: !mpiexec -n 4 python exercise4.py

Total working time: 12.553924199892208
Epochs:  10
```

```
In [51]: !mpiexec -n 4 python exercise4.py

Total working time: 16.029153099982068
Epochs:  15
```

```
In [44]: !mpiexec -n 4 python exercise4.py

Total working time: 17.346846299944445
Epochs:  20
```

```
In [43]: !mpiexec -n 4 python exercise4.py

Total working time: 19.712426299927756
Epochs:  25
```



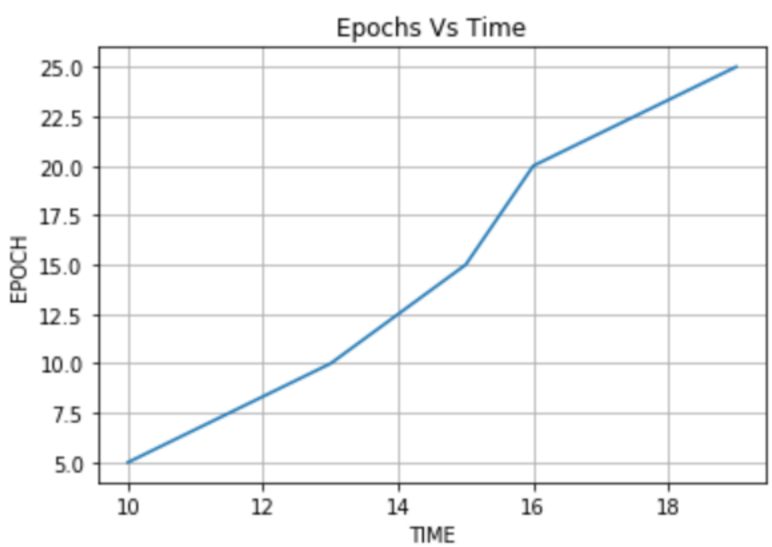
P=6:

```
[53]: !mpiexec -n 6 python exercise4.py

Total working time: 13.263573499862105
Epochs:  10
```

```
[50]: !mpiexec -n 6 python exercise4.py

Total working time: 15.36391680012457
Epochs:  15
```

```
[45]: !mpiexec -n 6 python exercise4.py

Total working time: 16.589137600036338
Epochs:  20
```

```
[47]: !mpiexec -n 6 python exercise4.py

Total working time: 19.090309400111437
Epochs:  25
```



P=8:

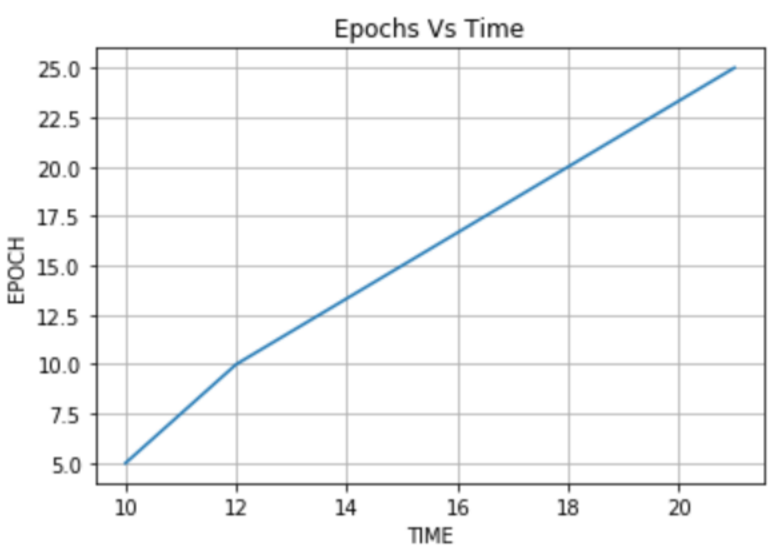
In [55]:
```
!mpiexec -n 8 python exercise4.py
```
```
Total working time: 10.662026700098068
Epochs:  5
```

In [54]:
```
!mpiexec -n 8 python exercise4.py
```
```
Total working time: 12.620306799886748
Epochs:  10
```

In [49]:
```
!mpiexec -n 8 python exercise4.py
```
```
Total working time: 15.433994800085202
Epochs:  15
```

In [46]:
```
!mpiexec -n 8 python exercise4.py
```
```
Total working time: 18.279678500024602
Epochs:  20
```

In [48]:
```
!mpiexec -n 8 python exercise4.py
```
```
Total working time: 21.63312439993024
Epochs:  25
```



All these above graphs shows that more epochs per unit time can be achieved using a parallel parallel processing.