**School of Electronic and Electrical Engineering**
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

**ELEC3875**

Project Report

# DYNAMIC ECONOMIC AND ENVIRONMENTAL POWER SYSTEM DISPATCH CONSIDERING PLUG-IN ELECTRIC VEHICLES

***Abdullah Essa***

| | |
|---|---|
| SID: 201256467 | Project No. 48 |
| **Supervisor:** Professor Kang Li | **Assessor:** Li Zhang |

**School of Electronic and Electrical Engineering**
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

## ELEC3875 Individual Engineering project

# Declaration of Academic Integrity

### *Plagiarism in University Assessments and the Presentation of Fraudulent or Fabricated Coursework*

***Plagiarism*** is defined as presenting someone else's work as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

***Fraudulent or fabricated coursework*** is defined as work, particularly reports of laboratory or practical work that is untrue and/or made up, submitted to satisfy the requirements of a University assessment, in whole or in part.

### Declaration:

- I have read the University Regulations on Plagiarism [1] and state that the work covered by this declaration is my own and does not contain any unacknowledged work from other sources.
- I confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes.
- I confirm that details of any mitigating circumstances or other matters which might have affected my performance and which I wish to bring to the attention of the examiners, have been submitted to the Student Support Office.

[1] Available on the School Student Intranet

Student Name: *Abdullah Essa* Project No. *48*

Signed: ..... Date: *6/05/2021*

## Abstract

This report proposes particle swarm optimization method for solving the dynamic economic and environmental dispatch problem in power systems via MATLAB to formulate a dispatch plan on different charge scenarios considering plug-in electric vehicles. Nonlinear characteristics of generation units such as ramp up and down rate limits, and non-smooth cost and emission functions are considered. The proposed method is compared with genetic algorithm optimization method in regards with ability to find global minimum and solution quality. The comparison results confirmed that the proposed particle swarm optimization method was capable of getting global minimum and better-quality solutions in economic dispatch problems. The dispatch plans for 10 generation units have been formulated using the proposed optimization method with off-peak scenario showing the lowest cost and emissions.

## Acknowledgements

I would like to thank University of Leeds and my sponsor Kuwait Cultural Office for giving me the opportunity and support to complete my studies. In addition, I thank my Individual Project supervisor Professor Kang Li and his third year PhD student Mingjia Yin each of whom provided continued patient advice and guidance throughout the research process. Finally, big thanks to my personal tutor Dr. Ahmed Lawey for his support and advice throughout my bachelor study years.

# Table of Contents

## List of Abbreviations

Definitions of acronyms used in text, listed in alphabetical order:

| | |
|---|---|
| **DE** | Differential Evolution |
| **DEED** | Dynamic Economic and Environmental Dispatch |
| **EP – SQP** | Evolutionary Programming with Sequential Quadratic Programming |
| **GA** | Genetic Algorithm |
| **HNN** | Hopfield neural network |
| **PEV** | Plug-in Electric Vehicle |
| **PSO** | Particle Swarm Optimization |
| **PSO – SQP** | Particle Swarm Optimization with Sequential Quadratic Programming |
| **SED** | Static Economic Dispatch |
| **PSO** | Particle Swarm Optimization |
| **EV** | Electric Vehicle |
| **PEV** | Plug-in Electric Vehicle |
| **BEV** | Battery Electric Vehicle |
| **HEV** | Hybrid Electric Vehicle |

# Chapter 1

## Introduction, Aim and Objectives

As awareness of environmental harmful pollutant emissions due to power plants is rising, it is becoming essential that emissions need to be taken into consideration in every operation that produces them. Thermal power plants that use fossil fuel as their primary supply of energy produce pollutants such as sulphur dioxide and nitrogen oxides [1, 3]. Knowingly, the use of electricity demand is ever-increasing, which also refers to an increase in means of supply, such as thermal power plants. This could breach the environmental emission regulations due to vast amounts of emissions that will create damaging environmental effects [14]. Now with the increasing number of plug-in electric vehicles due to raised environmental awareness that will impact the load demand with high penetration [1, 7], this is one candidate to look out for when creating a schedule for future load demands. Due to that fact, an increase in power generation dependent cost and pollutant emissions will occur. This is a problem that is usually solved by one of the main operational tasks, and that is to schedule the power output in accords with the load demand [1, 2, 4, 6, 8]. Dynamic economic and environmental power system dispatch can be used as a solution to a scheduling problem with motivation to minimize emissions and cost while meeting load demand [1, 4, 19, 21]. Thus, this project aims to decrease the cost and emissions of thermal power plant units when considering plug-in electric vehicles. This is to be done while managing system constraints (e.g., minimum & maximum power output, ramp rate constraints and optimization problems), balancing power production and load demand, as well as meeting plant operational requirements.

The objectives thus far:

- To understand the characteristics of thermal power plant generation operation, the economic and environment dispatch problem, and the impact of renewable generation. That is because we need to provide enough power to match the load with the aim of using less fuel consumption as possible (e.g., Principle of Equal Incremental Rate) without producing emissions that could badly affect the environment.
- Study the charging patterns as well as the charging/discharging scenarios of electric vehicles as it will be required to understand the all-out public usage of electric vehicles on the power systems.
- Start developing non-linear dynamic economic and environmental dispatch models. Study bio-inspired optimization algorithms to be applied on Matlab such as (Genetic algorithm).
- Choose a proper algorithm for the non-linear models that had been developed and propose a suitable dispatch plan for different charging/discharging scenarios.

# Chapter 2

# Literature Review

The content below is from multiple papers, thesis papers and books related to dynamic economic and environmental dispatch considering plug-in electric vehicles.

## 2.1 Net-zero Emissions

As industries and factories grew larger and more sophisticated, so did their greenhouse gas emissions proportionally. The UK government drafted a climate change act that is focused on at least a 100% reduction of greenhouse gas emissions by the year 2050 [23, 25, 26]. A 100% reduction of greenhouse gas emission is known as the net zero target.

## 2.2 Transport Electrification

Electrification is considered to be a great solution to avoid depending on oil to reduce the environmental impact [24]. Considering transport sector, it is by far responsible for about 16-25% of the global greenhouse gas emissions according to [24, 26]. Thus, transport electrification will offer a great alternative to the conventional fossil-fuel based transport technologies and reduces the greenhouse gas emissions.

Table (2.1), Some International EV target objectives from [28]:

| **Country** | **Targets** |
|---|---|
| United Kingdom | No target figures but policy to support EVs |
| China | 2011: 500000 annual production of EVs |
| USA | 2015: 1000000 PHEV stock |
| Austria | 2020: 100000 EVs deployed |
| Germany | 2020: 1000000 EVs deployed |
| Canada | 2018: 500000 EVs deployed |
| Ireland | 2020: 10% EV market share |
| Spain | 2014: 1000000 EVs deployed |
| Sweden | 2020: 600000 EVs deployed |

## 2.3 Types of Plug-in Electric Vehicles and Choice of State of Charge

Electric vehicles are known to be two groups, battery electric vehicles (BEVs) and hybrid electric vehicles (HEVs) [30]. The (BEVs) have the stored energy in the batteries provide 100% of the energy needed by the vehicle. The (BEV) has no other energy source, thus battery should be selected according to the range of the (BEV) and other requirements [29]. (HEVs) on the other hand are further broken down into three distinct groups; series hybrids, parallel hybrids, and combined hybrids [29]. What they have in common is that they have a higher range when comparing them to (BEVs) and have both electric motors and an internal combustion engine to drive the vehicle [29].

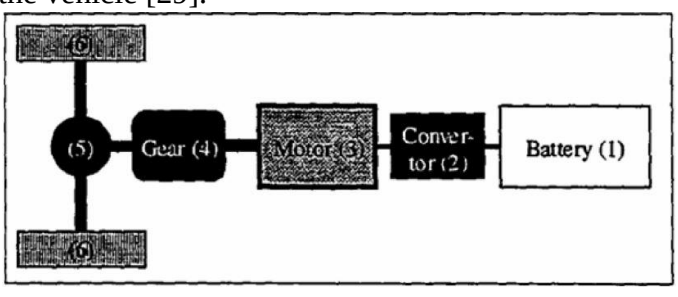


**Fig. 1** Structure of a single engine battery powered electric vehicle from [29].

According to figure (1), the drive system of (BEVs) consists of an electric motor, inverter, battery, gear box and a differential. Due to the fact that there is  also 100% dependence on battery only, the batteries will be having a higher amplitude than (HEVs) [31].

Series (HEVs) have the combustion engine energy output provide power to the electric generator that drives the electric motor [29]. Any additional energy is fed and stored in the vehicle battery. Parallel (HEVs) have both the electric motor and the combustion engine connected to the transmission system [29]. The vehicle can move with electric motor alone, both electric motor and combustion engine or combustion engine alone [29]. The combustion engine can also charge the batteries during [29]. Combined (HEVs) can use both combustion engine and electric motors with different ratios, completely combustion mode or completely electric mode with separate transmission path [29].

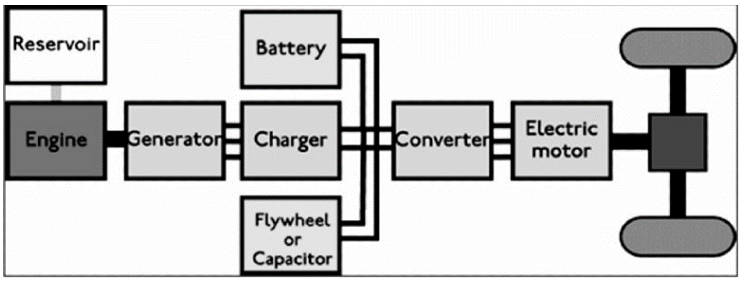


**Fig. 2** Structure of a series hybrid electric vehicle from [29].

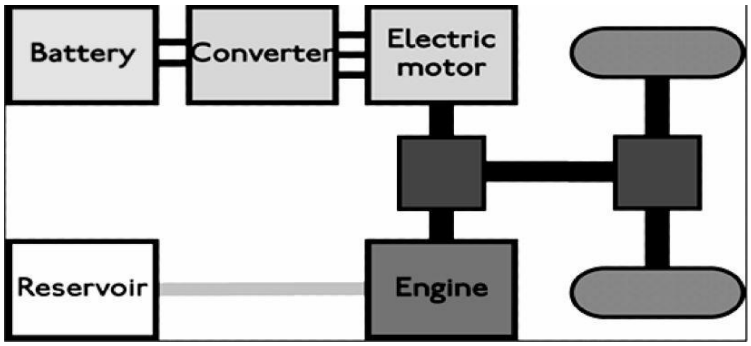


**Fig. 3** Structure of a parallel hybrid electric vehicle from [29].

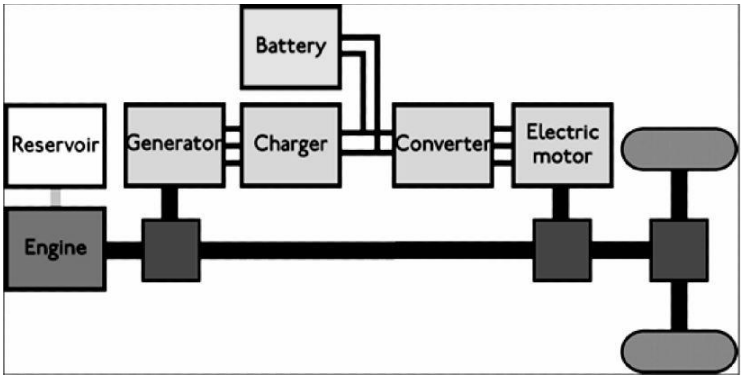


**Fig. 4** Structure of a combined hybrid electric vehicle from [29].

2.3.1 State of Charge

State of charge is the battery's current energy stored depending per its battery capacity (C) given by:

$$SOC = \left(\frac{E_C}{C} - 1\right) * 100 \quad (1)$$

Where ($E_C$) is the needed energy to charge the battery, ($C$) is the battery capacity in kWh [32].
The assumed SOC taken from [34] which explains how the mean daily energy remaining (i.e initial SoC) at the end of a day trip is 50% and 20% in winter and summer, respectively. For this reason, a pessimistic initial SoC of 20% is chosen. 80% SoC upon disconnection from charging will be considered as data from [34] shows that disconnection at around 80% has the highest probability.

2.3.2 Nissan Altra (BEV)

The energy needed to charge an electric vehicle will depend on what type of battery it is utilising. There are many types of batteries used to power EVs, those can be lithium ion, Lead-acid, and Nickel Metal Hydride (NiMH) [32]. For all the cases that will be explored in this project, Lithium-ion batteries will be used as it is more popular for driving PEVs compared to other battery types. For this report, the electric vehicle problem data will consider Nissan Altra as the test vehicle which has a capacity of 29.07kWh and a 5-hour charge time [32, 33]. Thus, it is required to examine different charging scenario percentages to see which are more economically and environmentally friendly to charge such battery.

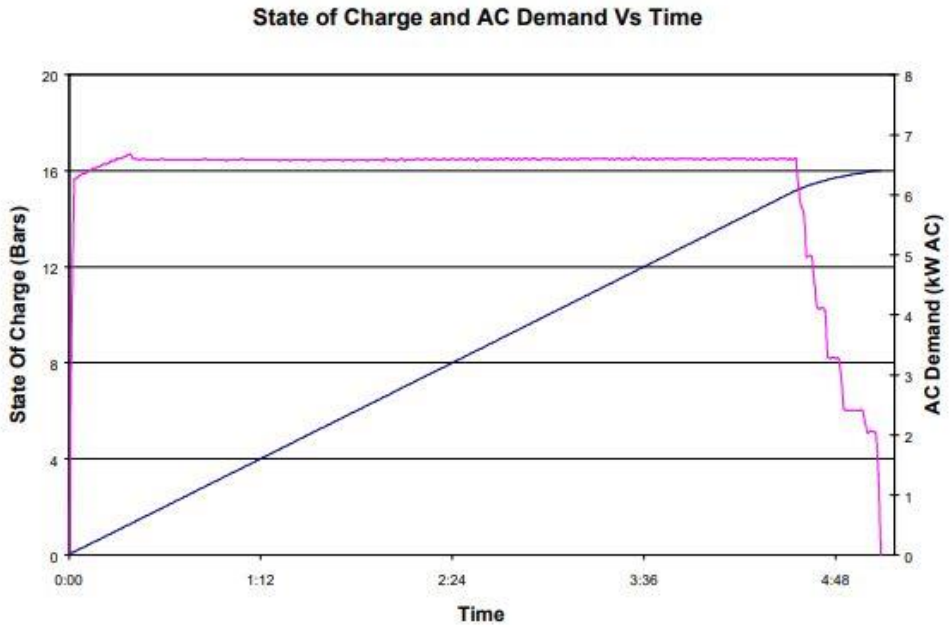


**Fig. 5** Plot of charge demand over a time period for Nissan Altra from [33].

The PEV charging profile shows how it charges for about 22.3% per hour for 4-hours and about 10.8% for the remaining hour until SOC is at 100%.

2.4 Plug-in Electric Vehicles Charging Scenarios

Many electric vehicles will be charging at the same time after their roll-out; each is consuming several (kW) or hundreds of (kW) power. This means that they are modelled as a single load with varied load demand behaviors. Such behaviors refer to the different charging scenarios taken from [1].

In this project the charging scenarios of plug-in electric vehicles that will be investigated are four. EPRI profile, off-peak profile, peak profile, and stochastic profile that takes into account the uncertain behavior of drivers when charging, random times during day and night.

Table (2.2), Showing the Peak charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 07:00-12:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 13:00-18:00 | 18.5% | 18.5% | 18.5% | 18.5% | 9% | 9% |
| 19:00-24:00 | 4% | 4% | 0% | 0% | 0% | 0% |

Table (2.3), Showing the Off-peak charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 18.5% | 18.5% | 9% | 9% | 4% | 4% |
| 07:00-12:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 13:00-18:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 19:00-24:00 | 0% | 0% | 0% | 0% | 18.5% | 18.5% |

Table (2.4), Showing the EPRI charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 10% | 10% | 9.5% | 7% | 5% | 3% |
| 07:00-12:00 | 1% | 0.3% | 0.3% | 1.3% | 2.1% | 2.1% |
| 13:00-18:00 | 2.1% | 2.1% | 2.1% | 1% | 0.5% | 0.5% |
| 19:00-24:00 | 1.6% | 3.6% | 5.4% | 9.5% | 10% | 10% |

Table (2.5), Showing the Stochastic charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 5.7% | 4.9% | 4.8% | 2.4% | 2.6% | 9.7% |
| 07:00-12:00 | 8.7% | 4.8% | 1.1% | 3.2% | 2.1% | 5.7% |
| 13:00-18:00 | 3.8% | 2.2% | 2.1% | 6.1% | 3.2% | 2.2% |
| 19:00-24:00 | 2.8% | 2.2% | 5.5% | 2.5% | 3.5% | 8.2% |

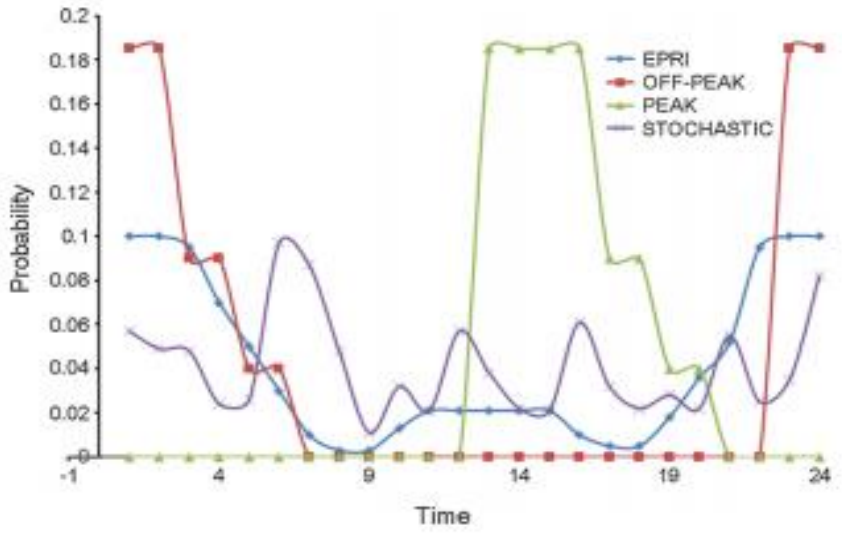


**Fig. 6** Probability plot of power supplied at four different charging scenarios with PEV load distribution from [1]

2.5 Plug-In electric vehicles impact on power demand

Plug-in electric vehicles household simultaneous chargers are power rated 20kW, and superchargers are power rated at 120kW [1]. Such demand will produce huge ripples or spikes in load profile of demand. The spikes are dangerous as generating units have ramp constraints that may not satisfy large sudden demands [1]. This is avoided by good scheduling techniques which is best done via one of many dispatch optimization methods [1].

## 2.6 Static Economic Dispatch (SED)

The need to schedule the generation units' output to meet the required load demand at minimum cost while satisfying system requirements is the primary goal of SED. Considering the problem formulation and objectives of economic dispatch SED part of DEED, the SED problem is non multi-objective problem that involves a fixed load demand at a chosen time stamp [1, 4, 6] given as:

$$\min F = \sum_{i=1}^{N} a_i + b_i P_{Gi} + c_i P_{Gi}^2 \tag{2}$$

such that

$$\sum_{i=1}^{N} P_{Gi} = P_D \tag{3}$$

Where N is the number of generation units $a_i$, $b_i$, $c_i$ are cost curve coefficients of the i[th] generation unit while $P_{Gi}$ represents the output power of the i[th] generation unit [1, 4, 6] with $P_D$ being the power demand.

System requirement satisfaction as in constraints that we need to consider for ED with regards to the project expectations includes power output limits [1, 6] given as:

$$P_{Gimin} \leq P_{Gi} \leq P_{Gimax} \ (i = 1,2,\ldots,N) \tag{4}$$

## 2.7 Dynamic Economic Environmental Dispatch (DEED)

DEED is a form of optimization problems/decision-making processes. In our case, it is a constrained optimization problem where the objective is to find the most economical and environmental schedule of generating units while satisfying load demand and operation constraints. The term dynamic is used because we consider various loads at different times. This is considered as a multi-objective problem [1] given as:

$$\min F = wF_1 + (1 - w)F_2 \tag{5}$$

The $w$ stands for the weighting factor as a constant value between 0 and 1 [1], helps choose which part of EED has a higher priority. $F_1$ and $F_2$ stand for the non-linear equations for dynamic economic dispatch and dynamic environmental dispatch respectively [1]. $\min F$ denotes the minimum overall cost [1, 4, 6].

$F_1$ given as [1]:

$$F_1 = \sum_{t=1}^{T} \sum_{i=1}^{N} a_i + b_i P_{Gi,t} + c_i P_{Gi,t}^2 + |d_i \sin(e_i(P_{Gimin} - P_{Gi,t}))| \tag{6}$$

Where N is the number of generation units. $a_i$, $b_i$, $c_i$ are cost curve coefficients of the i[th] generation unit while $P_{Gi}$ represents the output power of the i[th] generation unit [1, 4, 6]. Consideration of coefficients

$d_i$ and $e_i$ are for valve point effects of thermal generation units which evaluate the ripples in the cost curve [1, 20]. $T$ denotes the time intervals in a given load profile. The DEED system constraints to consider are power output limits and ramp constraints [1].

Power output constraints:

$$P_{Gimin} \leq P_{Gi,t} \leq P_{Gimax} \ (i = 1,2,\ldots,N) \tag{7}$$

The power output of each generator must be within constraints of the generator power output limits.

Ramp rate constraints:

$$\begin{aligned} P_{Gi,t} - P_{Gi,t-1} &\leq UR_i \\ P_{Gi,t-1} - P_{Gi,t} &\geq DR_i \end{aligned} \tag{8}$$

Consideration of ramp rate constraints because thermal power generation units cannot dramatically change within two adjacent intervals, as stated in [1, 4, 6]. $UR_i$ stands for the ramp-up constraint of a thermal generator and $DR_i$ stands for the ramp-down constraint of a thermal generator. Meaning that the power output of a given generator at interval $t$ must consider what power it was outputting at interval $t-1$ to be within the ramp-up and down constraints [1].

$F_2$ given as [1]:

$$F_2 = \sum_{t=1}^{T}\sum_{i=1}^{N} \alpha_i + \beta_i P_{Gi,t} + \gamma_i P_{Gi,t}^2 + \eta_i \exp(\delta_i P_{Gi,t}) \tag{9}$$

Where $\alpha_i$, $\beta_i$, $\gamma_i$, $\eta_i$ and $\delta_i$ represent the i$^{th}$ thermal generation units' emission coefficients. The goal is to minimize the emitted pollutants that have Sulphur dioxide and nitrogen oxides [1, 20].

Thus, when including these requirements with economic dispatch, a more sophisticated optimization method to look through the different demands for different times on a given load profile is needed.

2.8 Dispatch solving methods

Dispatch solving methods are of many types used for many different sets of problems. These solving methods can be classed into three primary categories. Each category has its pros and cons. The solving methods can be heuristic, meta-heuristic or hybrid methods. Some advantages of heuristic methods are not needing to specify problem parameters [19] and are fast to compute. Disadvantages of heuristic methods, however, are that since they are fast to compute this means they will converge pretty fast creating a very sensitive dependence on initial values [19] Heuristic methods such as Lagrange relaxation [2, 19], gradient projection method [19] and lambda iterative method [2, 19] can be used to solve smooth convex cost functions. With the non-convex cost function of a DEED, they fail to achieve a global optimal solution [19]. In reality, dispatch models are much more complex and require the formulation of dispatch models for N amount of generation units each with their constraints. Meta-heuristic methods involve differential evolution (DE) [10, 19], Hopfield neural network (HNN) [19], particle swarm optimization (PSO) [2, 9, 16, 19] and genetic algorithm (GA) [2, 5, 6, 19]. These methods can seek for the global optimal solution with or without constraints that are applied to the cost function. The advantage is that they do not have a sensitive dependence on initial values but do suffer from long compute times as well as having to specify many problem parameters [5, 9, 11, 15, 19, 21]. Hybrid methods allow the combination of different solving methods to have the best of their features combined into one method. Hybrid methods have proven their effectiveness in problems related to DEED using particle swarm optimization combined with sequential

quadratic programming (PSO – SQP) [2, 16, 19] and evolutionary programming combined with sequential quadratic programming (EP - SQP) [19].

### 2.7.1 Quadratic Programming (Quadprog)

Quadprog is a heuristic quadratic programing function [18] on Matlab That finds the minimum cost of a given quadratic function and its problem parameters.

Equation given as [17]:

$$Cost_{min} = \frac{1}{2} * x^T * H * x + f^T * x \tag{10}$$

Considering equation (1), x represents committed generation unit output power $P_i$ from cost curve function. H is a matrix of size $(ixi)$ and the cost coefficient (c) of $P_i$ is at $H_{ii}$ [17]. The variable f is a matrix with size $(1xi)$ and cost coefficient (b) of $P_i$ is at $f_{1i}$ [17].

The minimum cost is controlled by the matrices of inequality constraints, equality constraints, upper bound and lower bound of the operational characteristics of generation units [17].

When used in SED it returns the minimum of each generation unit cost from a given cost function while considering their output power constraints [17]. The output result of total minimum cost of this quadprog function does not consider the $(a)$ constant of each generation unit cost curve function, meaning it will be added separately after performing the function.

Matlab function written as:

$$[X, J] = quadprog(H, f, A_{inequality}, B_{inequality}, A_{equality}, B_{equality}, UB, LB)$$

Where $A_{inequality}$ represents the matrix of coefficients of the provided inequalities and $B_{inequality}$ represents the objective matrix of inequalities. $A_{equality}$ represents the matrix of coefficients of the provided equality constraints and $B_{equality}$ represents the results of each equality constraint. The upper bound and lower bound of each generation unit is represented by the variables $UB$ and $LB$ respectively. X represents the final variable that is multiplied by the cost coefficients, meaning the power output of committed generation unit. J represents the minimum total cost without considering the addition of $(a)$ constant of each generation unit cost curve function.

### 2.7.2 Genetic Algorithm

Genetic algorithm is a method that is part of the meta-heuristic methods [2, 5, 6, 19]. Meaning it has certain advantages that allow for the chance to achieve global optimal solution for a given DEED problem. The genetic algorithm that is being looked at is the binary encoded genetic algorithm. This optimization method uses bio-inspired steps to reach global optimal solution. These steps include random population initialization and repetitive use of finding fitness, crossover, and mutation [2, 5, 6, 19, 20]. Many methods are used to help choose the parents for crossover such as rank and tournament method [20, 22]. For a crossover, itself has many methods as well, but the single-point crossover is the method that is widely used [5, 20]. Then mutation is when randomly choosing an encoded binary bit and flipping it [20]. These are done iteratively, wherein which the new offspring will share some of the parents' characteristic [2, 5, 20]. Depending on whether minimum or maximum possible result is what is needed, the algorithm can be formulated accordingly, and through generations of populations, the global optimal solution is searched for and achieved when the fitness converges [2, 20].

### 2.7.2 Particle Swarm Optimization

Particle swarm optimization is part of the metaheuristic optimization methods, it is inspired by the natural behaviour of bird flocking and fish schooling [21]. The method contains particles with multiple dimensions if need be, to search through a search space via a computed velocity until a position is reached which represents the solution [12, 16, 21]. The position of the particles and their dimensions is updated on each iteration via the computed velocity until a global minimum is reached [12, 16, 21].

- There are (d) dimensions for each particle.
- For each dimension there is a velocity. Thus, a vector of velocities with size (dx1) is required.
- For each dimension there is a position. Thus, a vector of positions with size (dx1) is required.
- For each velocity and position we determine the particles' best dimension value and stored as (gbest).
- The velocities of the particles are tuned depending on the requirement (whether more exploration is needed or obtain better and better solutions in the neighbourhood of the present solution).
- The velocity is tuned via a linearly adaptable inertia weight (w).

The equation that defines the value of the velocity and position of each dimension depending on present and past iteration from is given by [21],

$$v_{id} = w * {v_{id}}^{k} + C_1 * (Pbest_{id} - {x_{id}}^{k}) + C_2(gbest_d - {x_{id}}^{k}) \tag{11}$$

$$\begin{gathered} {x_{id}}^{(k+1)} = {x_{id}}^{k} + {v_{id}}^{(k+1)} \\ i = 1, 2, \dots, N_p \\ d = 1, 2, \dots, N_g \end{gathered} \tag{12}$$

- $N_g$ stands for the total number of dimensions.
- $N_p$ stands for the population size.
- $Pbest_{id}$ stands for the best position so far for particle (i) at dimension (d).
- $gbest_d$ stands for the best among all of the particle (i) dimensions.
- $v_{id}$ stands for the velocity of particle (i) at dimension (d).
- $x_{id}$ stands for the position of particle (i) at dimension (d).
- $C_1$ and $C_2$ are randomly chosen constants that provide relative stochastic weighting of the deviation from best self-performance of the particle itself.

The equation for linearly adaptable inertia weight that tunes the velocity is given by [21],

$$w = \frac{w_{max} - w_{min}}{iter_{max}} * iter \tag{13}$$

- $iter_{max}$ stands for the maximum number of iterations.
- $iter$ stands for the current iteration.
- $w_{max}$ & $w_{min}$ stand for maximum and minimum weight respectively which alter the velocity depending on the current iteration.

# Chapter 3

## Formulating Binary Encoded Genetic Algorithm Optimization Method

At this stage of the project, an attempt to formulate the genetic algorithm method is made as a static economic dispatch solving method. The upcoming steps are to help understand how the results in the next chapter came to be using genetic algorithm optimization method code in (Appendix B).

3.1 Population creation and fitness function

Each population member is of a different row, and the number of columns it utilizes is of the chosen bit resolution. The number of columns for a given amount of (i) generation units is given by:

$$N_{columns} = (i-1) * n \tag{14}$$

Where (n) represents the chosen bit resolution. The number of rows would represent the total population used for the algorithm. The genetic algorithm code would be able to decode any size matrix of random binary initial population solutions and give the real values for any (i) amount of generation units with their power output constraint. Uses this equation [2]:

$$\begin{gathered} DecodedValue \rightarrow (Binary \rightarrow Decimal)\ conversion \\ RealValue = \frac{P_{Gi\,max} - P_{Gi\,min}}{2^n - 1} * DecodedValue \end{gathered} \tag{15}$$

The fitness of each population individual is achieved using the real values inputted into the formulated fitness function. Fitness function used here depends on a fuel cost function which is the real output power values of the generating units inputted into their input-output characteristic equation multiplied by the cost per MW power output and the demand we are trying to match [2]. The total fuel cost function for each thermal generation unit is given by equation (2).

$P_{GN}$ is the reference unit given as:

$$P_{GN} = P_D - \sum_{i=1}^{N-1} P_{Gi} = P_D - P_{Gi} - P_{Gi+1} \ldots - P_{N-1} \tag{16}$$

Since the i[th] generation unit is considered as the reference unit [2], the operational system constraints are implemented via the penalty factor equation [2] given by:

$$h = h_1(P_{GNmax} - (P_{GN}))^2 + h_2((P_{GN}) - P_{GNmin})^2 \tag{17}$$

Here, adding penalty factors to the violation of slack unit power output. Different combinations of $h_1$ & $h_2$ values have been tested to see which are best fit for a given problem. $P_{GNmax}$ & $P_{GNmin}$ represent the upper and lower output power constraints of the reference unit. With numerous testing, the best choice of penalty factors is $h_1 = 10$ and $h_2 = 100$.

Then Fitness function is given by [2]:

$$F_{fitness} = F + h \tag{18}$$

Where $F$ represents the outcomes from equation (1) per generation unit and h representing the outcomes of penalty factor equation from equation (17).

### 3.2 Choosing parents

Tournament method to choose the best possible parents for cross-over has been used. This tournament method focuses on choosing random population individuals' fitness and have it compete with other population individuals' fitness at least twice [22]. The winner is chosen on the basis of who has the lower fitness value.
The number of winners will equal the number of population individuals chosen at the start of the algorithm. We choose those winners to become parents in cross-over.

### 3.3 Crossover

We use the winners of the tournament to go through the cross-over code. Parents will have a probability of crossover set at the beginning of the GA code.
The cross-over technique used is (Single-Point Cross-Over), which means that we choose a binary position (point) within both parent solutions. Then swap the lower significant bits after the single point cross-over bit between the two parents [2, 5, 20].

Table (3.1) showing 4-bit resolution parents' example for cross-over with single point cross-over at bit 2:

| | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|
| Parent 1 | 1 | 0 | 0 | 1 |
| Parent 2 | 1 | 1 | 1 | 1 |
| Offspring 1 | 1 | 0 | 1 | 1 |
| Offspring 2 | 1 | 1 | 0 | 1 |

### 3.4 Mutation

We use the solutions of all offspring and mutate each bit depending on the probability of mutation. This is implemented by visiting each element of a matrix and flipping the bit [2, 20]. This is done in an attempt to help visit all possible solutions via a given population offspring.

### 3.5 Choice of next generation candidates

To get next generation candidates it is required to combine solutions of both offspring and parent, sort their fitness and choose the best candidates to become the new population for the next generation [2, 5, 6, 20]. This is done depending on the chosen iterations for the algorithm. Once chosen iterations are met, the final result will be the minimum fitness out of all the population members.

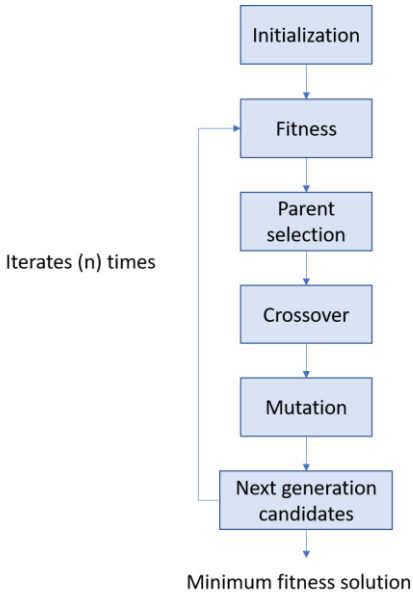


**Fig. 7** Implementation stages of the binary encoded genetic algorithm optimization method used for this project.

# Chapter 4

## Particle Swarm Optimization Method

This metaheuristic method contains particles with multiple dimensions if need be, to search through a search space via a computed velocity until a position is reached which represents the solution [12, 16, 21]. Below explains the steps taken to formulate the PSO code in appendix (C).

### 4.1 Initialization

During initialization, positions of each particle and particle dimension is generated randomly formulating a matrix size of (P, D) where (P) is the number of particles and (D) is the number of dimensions for each particle. They are generated while abiding the rules laid out by the constraints. This is performed before the first iteration for each hour power demand.

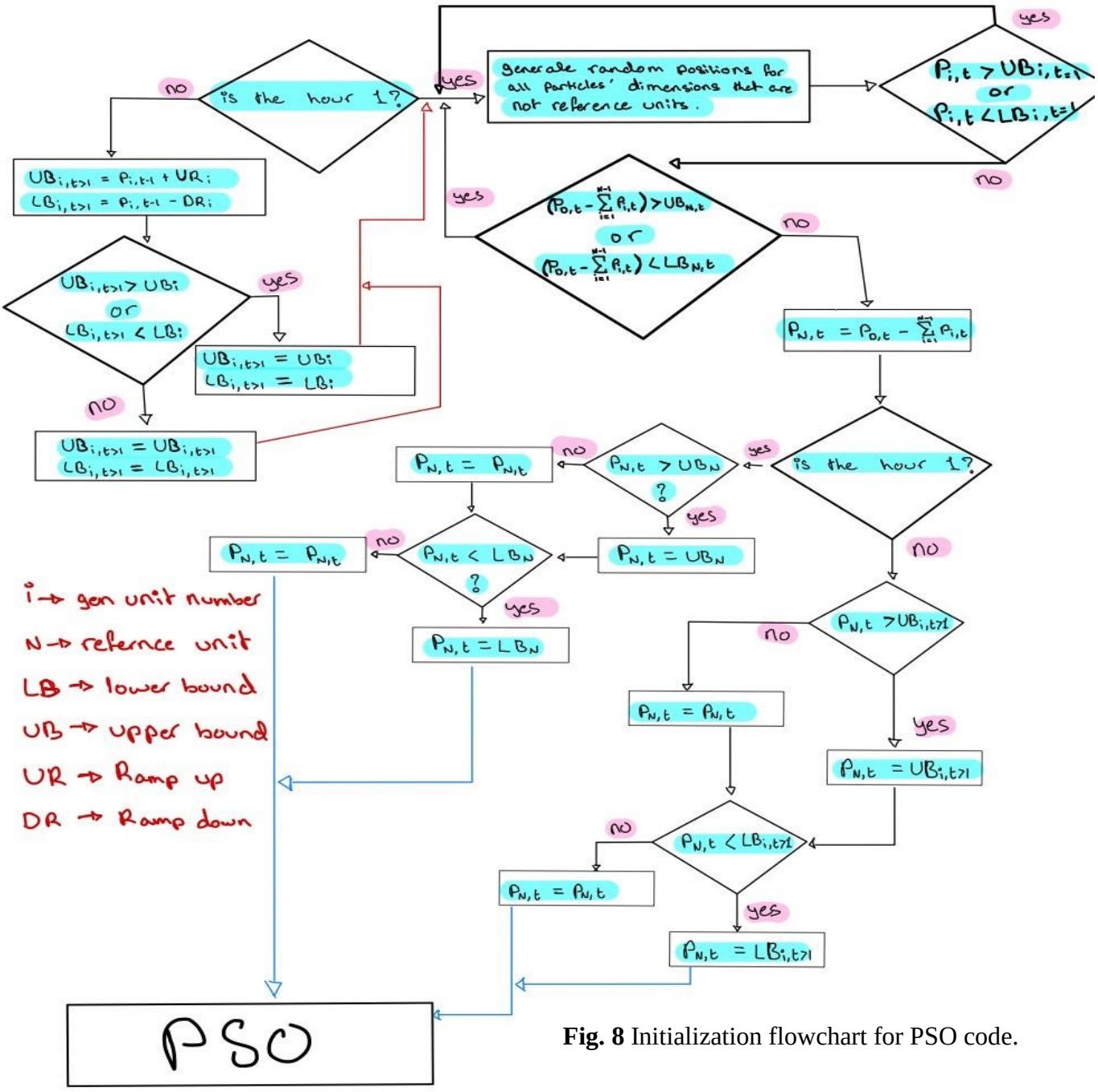


**Fig. 8** Initialization flowchart for PSO code.

As shown on figure (8), the randomly generated initial positions for the power demand hour (1) are only constrained by maximum and minimum output power as shown in equation (7). For the hours greater than one, it is shown on figure (8) that ramp up and down constraints shown in equation (8) are applied to further constrain the initial positions.

To allow reference generation unit to function properly, the initial positions for both cases above are constrained further by:

The positions are then recorded in an array to compute the fitness of each particle using equation (5). The best fitness is recorded as both personal best and global best. Initial velocities are generated randomly for each particle and its dimensions according to their limitations.

### 4.2 Particle Movement

Upon entering the first iteration for the PSO computations, the stored positions are updated via equation (12) and the velocity changes are done via equation (11). The values for constants (C1), (C2) and (w) are chosen accordingly after testing to achieve best global minimum. Where the ratio between dependence of particles' best record (C1) and the best record between all particles (C2) according to equation (11).

The fitness of each particle is calculated to find and store both personal best and global best. This is repeated multiple times depending on the set chosen number of iterations while abiding by the set constraint rules.

### 4.3 Choice of solution

The choice of solution is determined by the global best fitness after the last iteration. The global best fitness is used as reference to find the stored global best particle position. Then solution is used to compute both cost and emission.

# Chapter 5

# Method Comparison

### 5.1 Problem setup

The comparison analysis will use 5-units with operational characteristics taken from [21] and load demand of 500 MW.

Table (5.1), Table of Problem data for the 5-unit base problem from [21].

| Unit | $P_{max}$ (MW) | $P_{min}$ (MW) | UR (MW/h) | DR (MW/h) | a ($/h) | b ($/MWh) | c ($/MW$^2$h) |
|---|---|---|---|---|---|---|---|
| 1 | 75 | 10 | 30 | 30 | 25 | 2.0 | 0.0080 |
| 2 | 125 | 20 | 30 | 30 | 60 | 1.8 | 0.0030 |
| 3 | 175 | 30 | 40 | 40 | 100 | 2.1 | 0.0012 |
| 4 | 250 | 40 | 50 | 50 | 120 | 2.0 | 0.0010 |
| 5 | 300 | 50 | 50 | 50 | 40 | 1.8 | 0.0015 |

Table (5.2), Table of population and iterations for each optimization method used.

| Method | Population |
|---|---|
| Quadratic Programming | N/A |
| Genetic Algorithm | 10000 |
| Particle Swarm Optimisation | 10000 |

## 5.2 Quadratic Programming Analysis

Table (5.3), Table of results using the data from table (4.1) inputted in quadprog function on Matlab:

| | Average Power Output (MW) | Cost ($/h) |
|---|---|---|
| Generator 1 | 18.66197 | 65.11009529 |
| Generator 2 | 83.09859 | 230.2936 |
| Generator 3 | 82.74648 | 281.984 |
| Generator 4 | 149.29577 | 501.9054 |
| Generator 5 | 166.19718 | 380.58718 |

Total cost per hour = 1459.880202 $/h

### 5.2.1 Reviewing results

The results from quadratic programming show the power output for each generation unit and total cost considering the last unit as the reference unit. The average total cost per hour was found to be 1459.880202 $/h. The power output assignment of each committed unit via quadprog program being done without considering the ith generation unit as reference generation unit. Quadprog computes all the functions using H matrix without having the ability to manipulate what type of data gets to be fed into the last element of H matrix which represents the ith generation unit.

## 5.3 Genetic Algorithm Analysis

Table (5.4), containing genetic algorithm code parameters:

| | |
|---|---|
| Iterations | 100 |
| Population members | 10000 |
| Bit resolution | N/A |
| Generation units | 5 |
| Penalty factor $h_1$ | 1 |
| Penalty factor $h_2$ | 100 |
| Cross-over probability | 90% |
| Mutation probability | 10% |
| Power demand (MW) | 500 |

5.3.1 Testing different bit resolutions

Experimenting the effect of resolution bits on the fitness plot using data and parameters from table (1) and (3).

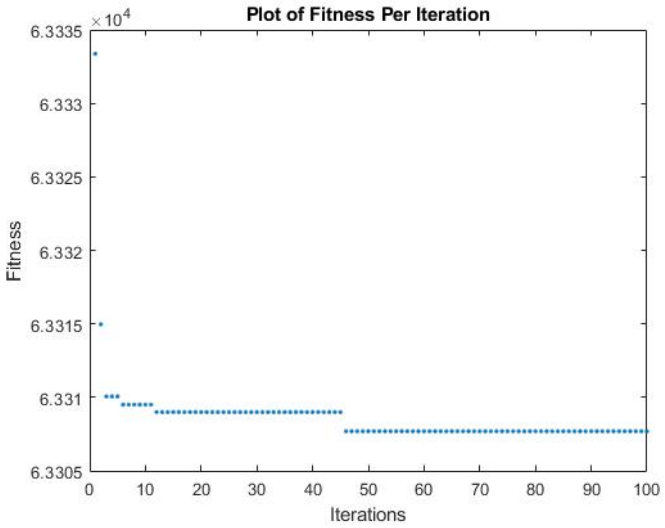


**Fig. 9** Plot of fitness against iteration number for a 4-bit resolution.

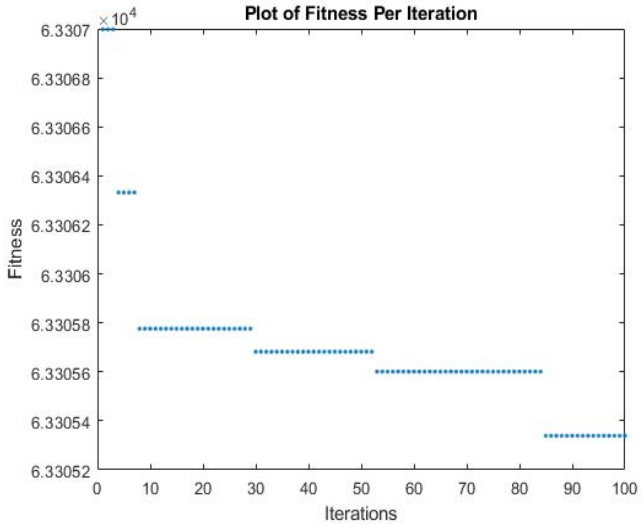


**Fig. 10** Plot of fitness against iteration number for an 8-bit

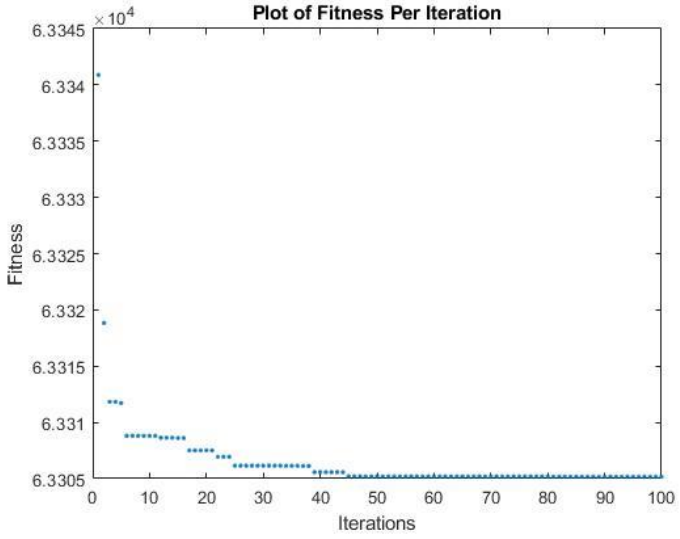


**Fig. 11** Plot of fitness against iteration number for a 16-bit

5.3.1.1 Reviewing results

Looking at the effect of bit resolution on fitness, it shows how only discrete values are assigned as the fitness over several iterations regardless of real-life value. It is a compromise between accuracy and computation time. Thus, due to results from case (1), case (2) and the experiment, it is deduced that a 16-bit resolution from case (3) is best to use for 10000 population members. Also, it shows that fitness function converges at different iterations depending on the problem parameters and initial random population values.

5.3.2 Applying optimization method on fixed load demand

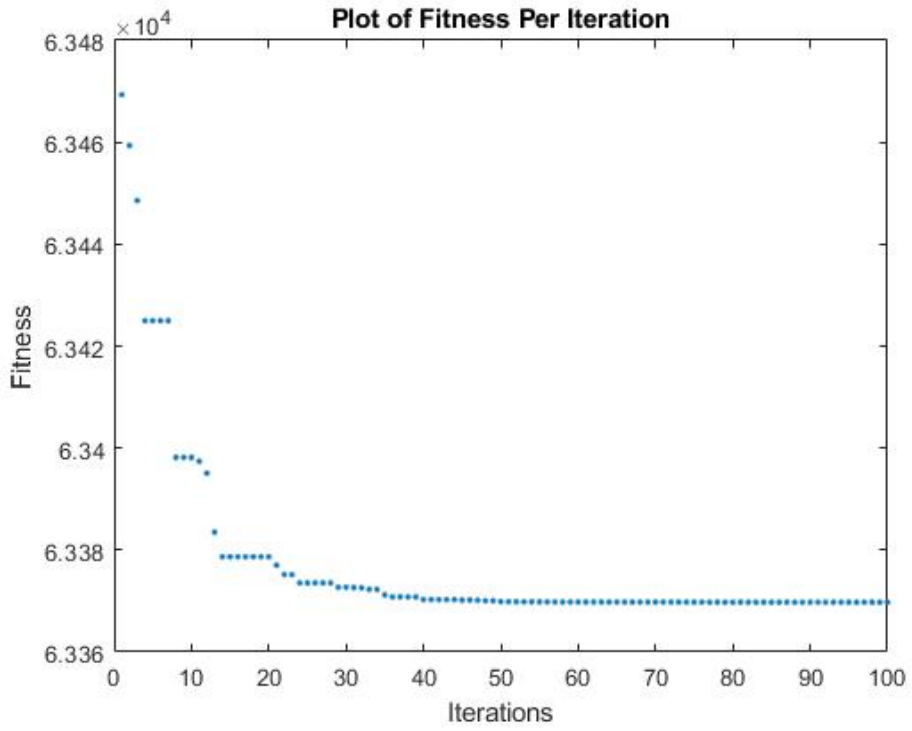


**Fig. 12** Plot of fitness against iteration number for one of the five algorithm runs for 500 MW demand. Using data and parameters from table (1) with 16-bit resolution.

Table (5.5), power output per generation unit and total cost using data of 5 algorithm runs:

| Run Number | Generator 1 MW | Generator 2 MW | Generator 3 MW | Generator 4 MW | Generator 5 MW | Total Cost $/h |
|---|---|---|---|---|---|---|
| 1 | 26.4536 | 98.5397 | 112.6966 | 209.8169 | 52.4932 | 1488.1 |
| 2 | 26.2493 | 98.7544 | 112.6744 | 209.8169 | 52.5050 | 1488.5 |
| 3 | 26.2304 | 98.7512 | 112.7010 | 209.8169 | 52.5005 | 1488.5 |
| 4 | 26.2413 | 98.7512 | 112.6966 | 209.8169 | 52.494 | 1488.6 |
| 5 | 25.6621 | 98.5397 | 112.6744 | 210.6276 | 52.4961 | 1490.7 |

Average total cost per hour = 1488.88 $/h

5.3.2.1 Reviewing results

The plots reach convergence at around 50 iterations for a population of 10000. The results from the binary encoded genetic algorithm show the power output for each committed generation unit with their cost considering the last generation unit as the reference unit. From five runs, the average total cost per hour was found to be 1488.88 $/h.

5.4 Particle Swarm Optimization Analysis

Table (5.6), containing Particle Swarm Optimisation code parameters:

| | |
|---|---|
| Iterations | 3000 |
| Particles | 10000 |
| C1 coefficient | N/A |
| C2 coefficient | N/A |
| w – inertia weight maximum | N/A |
| Power demand (MW) | 500 |

5.4.1 Changing acceleration constants

Experimenting the effect of different (C1) and (C2) coefficients on the fitness plot using data and parameters from table (1 & 6) and with inertia maximum set to (1) while iterations are set high to explore iterations needed to reach convergence.

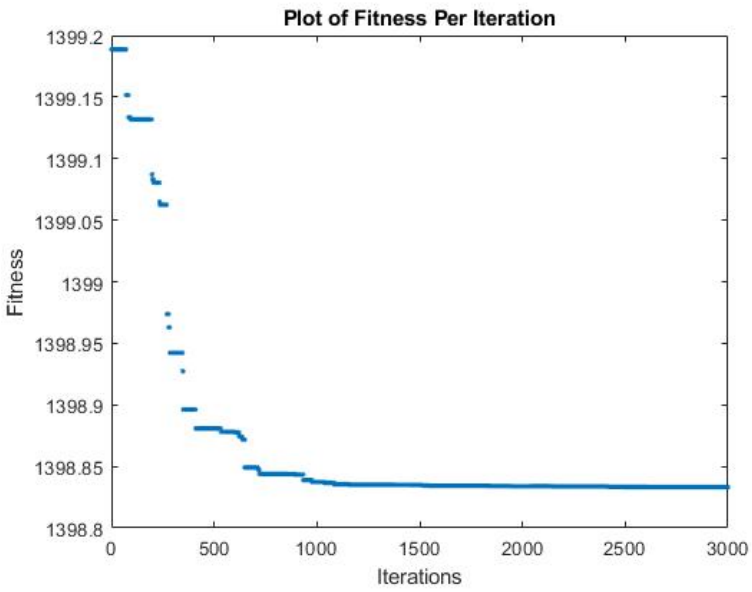


**Fig. 13** Plot of fitness against iteration number for (C1 = 1) and (C2 = 1).

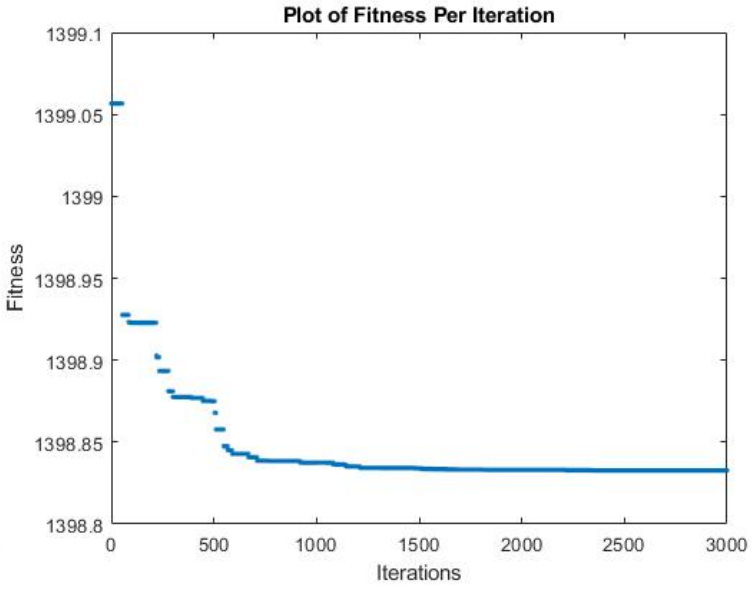


**Fig. 14** Plot of fitness against iteration number for (C1 = 1) and (C2 = 1.5).

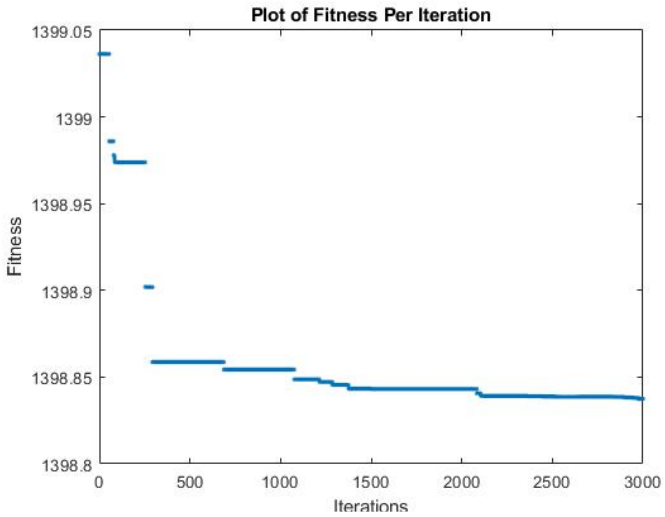


**Fig. 15** Plot of fitness against iteration number for (C1 = 1.5) and (C2 = 1).

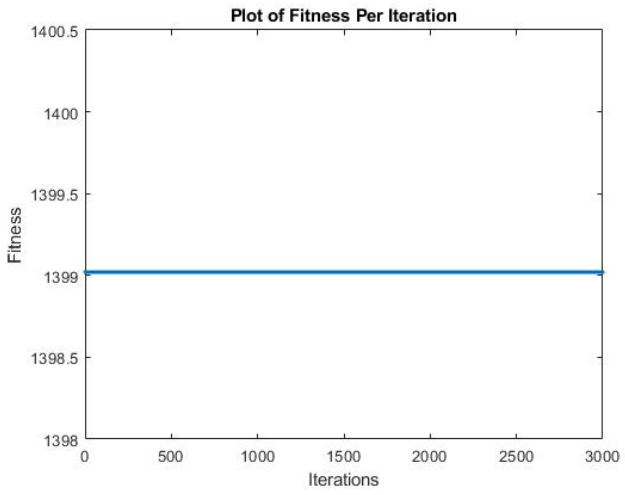


**Fig. 16** Plot of fitness against iteration number for (C1 = 0) and (C2 = 0).

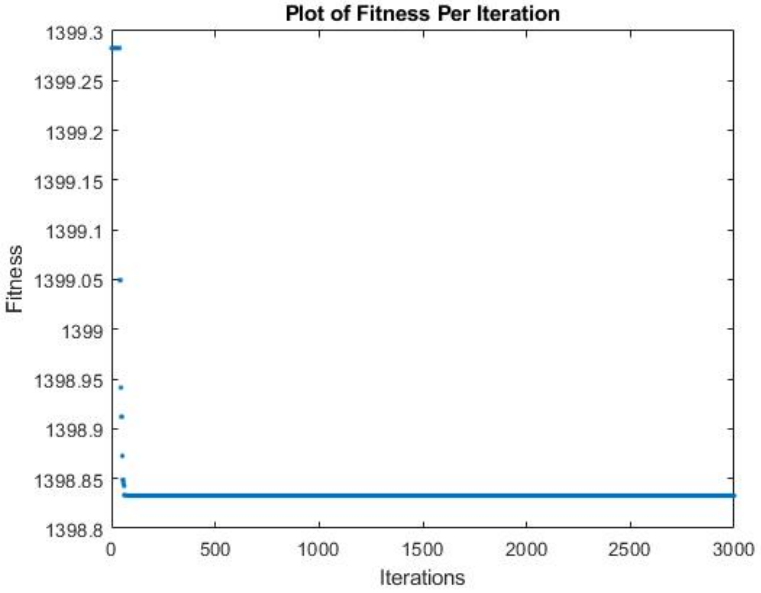


**Fig. 17** Plot of fitness against iteration number for (C1 = 0) and (C2 = 1).

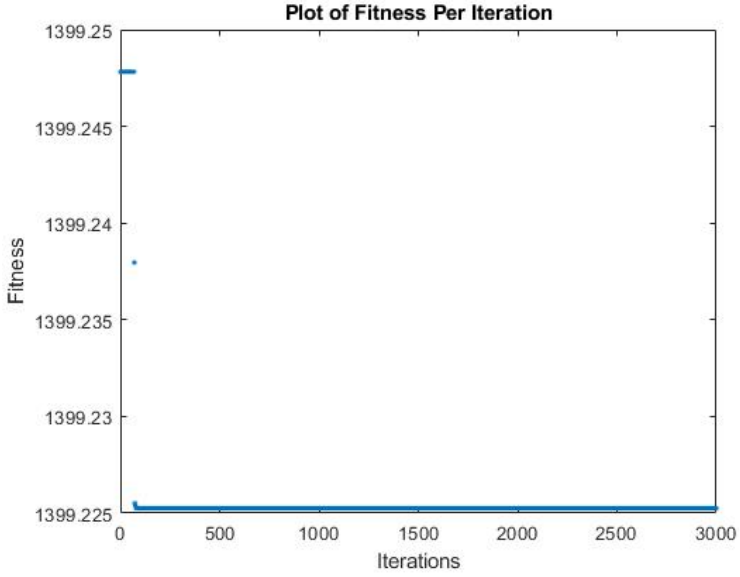


**Fig. 18** Plot of fitness against iteration number for (C1 = 1) and (C2 = 0).

5.4.1.1 Reviewing results

The (C1) constant represent the preference of search rather than communication between other particles, and the opposite meaning goes to the (C2) constant. Figure (16) shows that the fitness is at a stand-still since there is no search and communication between particles. Figures (14) and (15) show how with high number of particles it seems to be too exaggerated for (C1) searching, rather best fit for (C2) communicating with each other as search space occupied from the start is large reducing the number of iterations from over 3000 down to about 1600 iterations. Having them both set at the same priority which is seen on figure (1) reduces the iterations needed down to about 1500 iterations. Figures (17) and (18) show that when the priority of one of the coefficients is set to be unweighted (set to zero), the global minimum is found much faster. The iterations needed for convergence reduce to about 35 iterations for figure (17) and 45 iterations for figure (18). The fitness plots for figure (17) and (18) are better shown on figure (19) and (20) respectively.

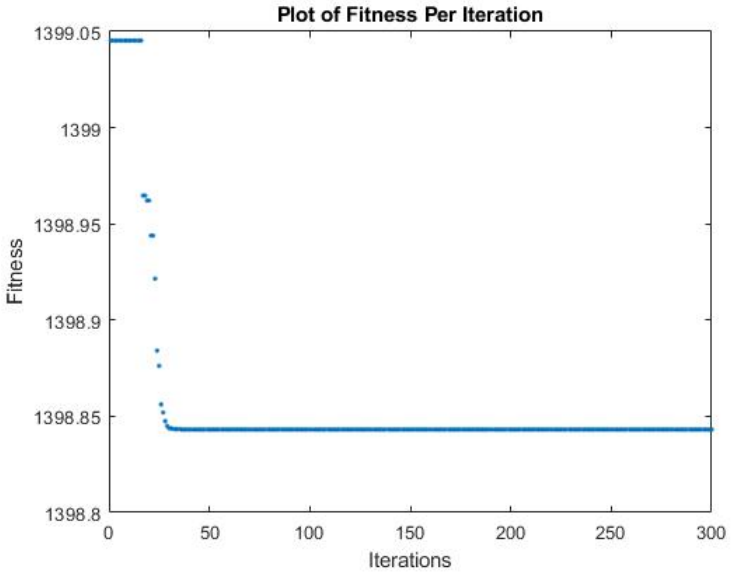


**Fig. 19** Plot of fitness against iteration number for (C1 = 0) and (C2 = 1).

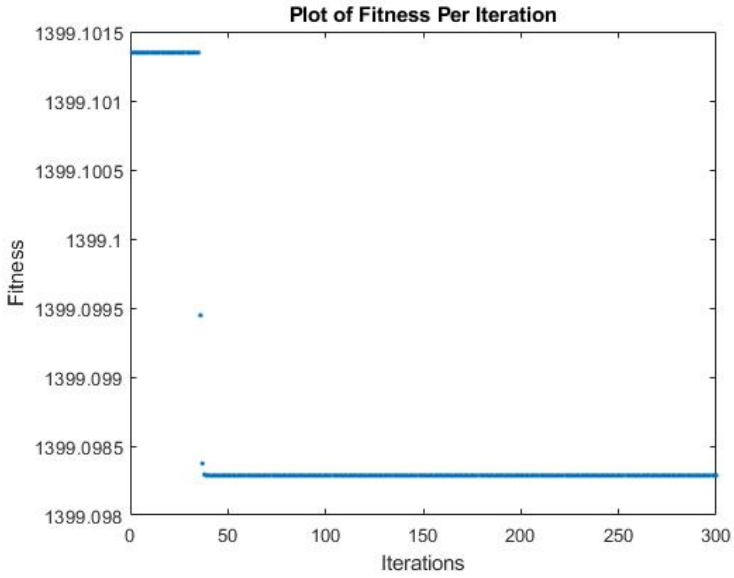


**Fig. 20** Plot of fitness against iteration number for (C1 = 1) and (C2 = 0).

5.4.2 Changing inertia weight

Experimenting the effect of different inertia weight maximum (w) on the fitness plot using data and parameters from table (1 & 6) and with (C1 = 0) and (C2 = 1) with iterations set to (300).

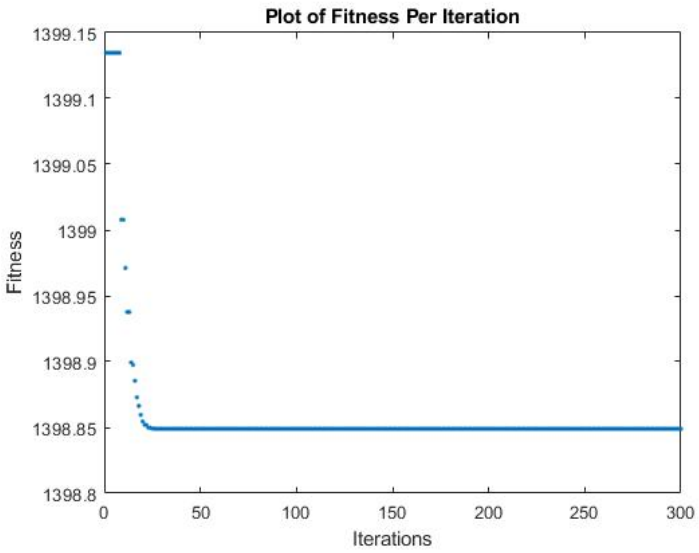


**Fig. 21** Plot of fitness against iteration number for (w = 0.1).

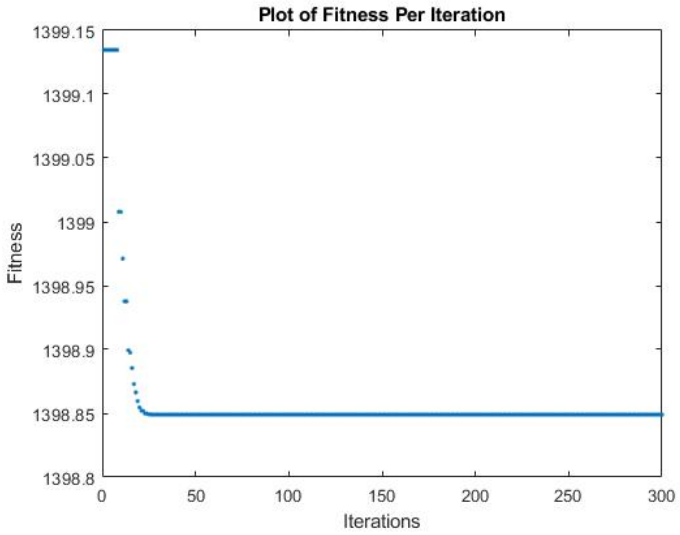


**Fig. 22** Plot of fitness against iteration number for (w = 0.5).

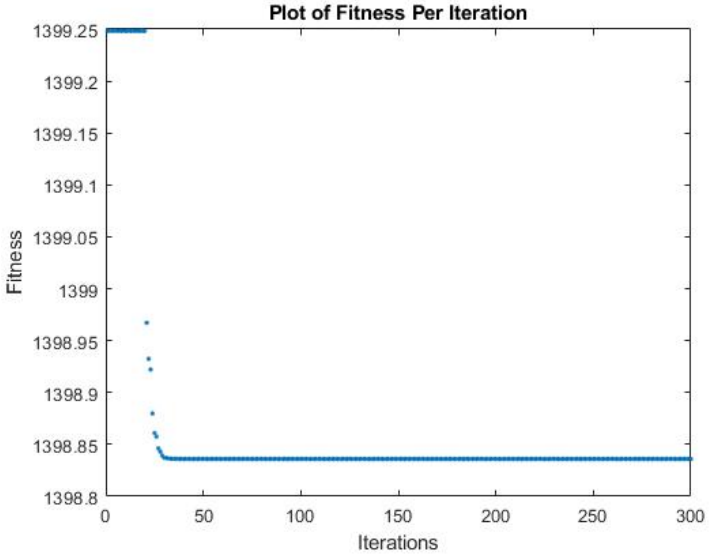


**Fig. 23** Plot of fitness against iteration number for (w = 1).

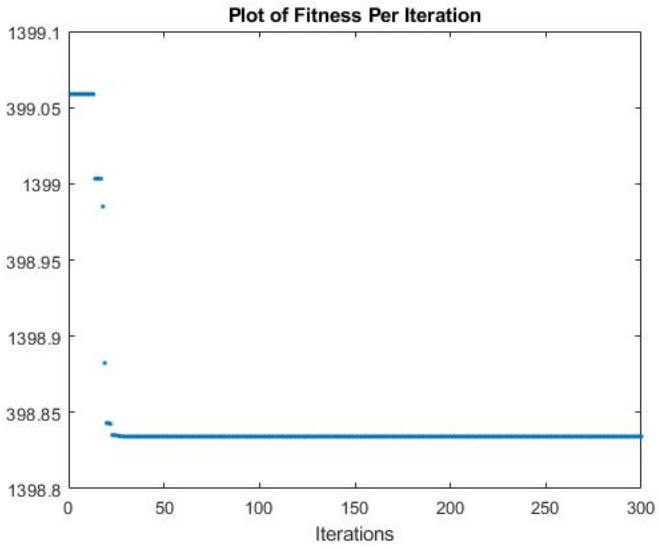


**Fig. 24** Plot of fitness against iteration number for (w = 0.8).

5.4.2.1 Reviewing results

Inertia weight (w) aids the particles for searching, having a greater maximum would persuade the particles to slightly prioritize search over communication which would increase the iterations needed for convergence. All inertia weight values produce about the same result of number of iterations needed to reach convergence. This is because coefficient (C1) was set to be (0) meaning velocity to change particle position is almost negligible.

5.4.3 Applying optimization method on fixed load demand

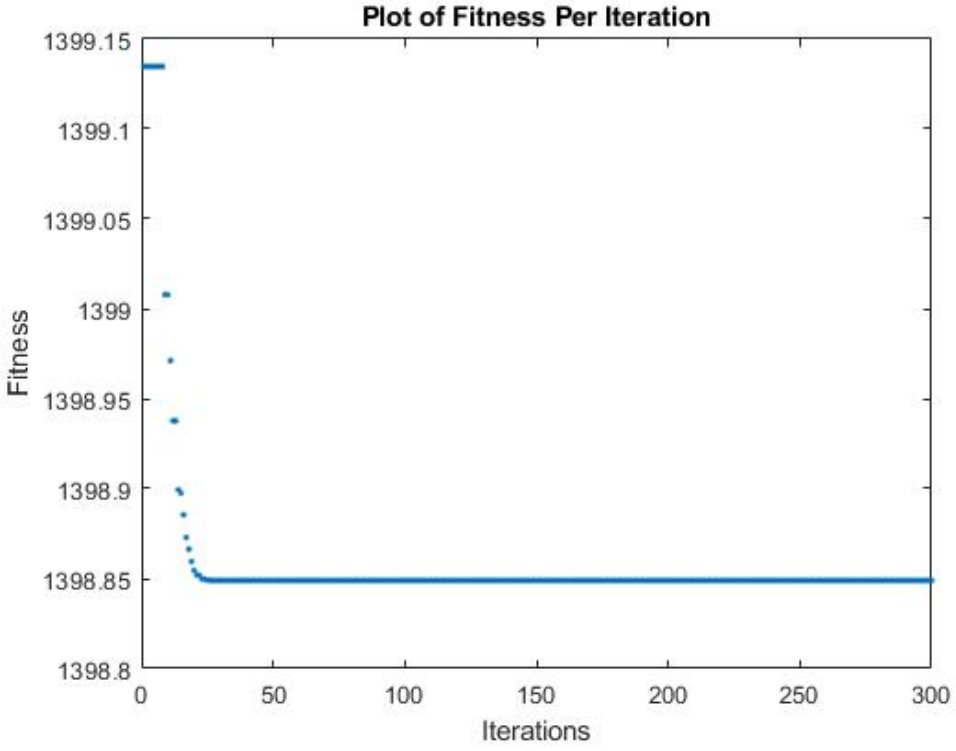


**Fig. 25** Plot of fitness against iteration number for data in table (1 & 6) with (w = 0.5), (C1 = 0), (C2 = 1), and 300 iterations.

Table (5.7), power output per generation unit and total cost using data of 5 algorithm runs:

| Run Number | Generator 1 MW | Generator 2 MW | Generator 3 MW | Generator 4 MW | Generator 5 MW | Total Cost $/h |
|---|---|---|---|---|---|---|
| 1 | 18.6589 | 83.0505 | 82.7581 | 149.2187 | 166.3037 | 1398.8 |
| 2 | 18.6917 | 83.0199 | 82.4116 | 149.5448 | 166.3219 | 1398.8 |
| 3 | 18.6619 | 83.1271 | 82.9899 | 149.1822 | 166.0289 | 1398.8 |
| 4 | 18.6338 | 83.0449 | 83.0301 | 149.0642 | 166.217 | 1398.8 |
| 5 | 18.7697 | 83.1684 | 82.7448 | 149.289 | 166.0181 | 1398.8 |

Average total cost per hour = 1398.8 $/h

5.4.3.1 Reviewing results

The plots for PSO reach convergence at around 25 iterations for a population of 10000. The results from particle swarm optimization show the power output for each generation unit and total cost considering the last unit as the reference unit. From five runs, the average total cost per hour was found to be 1398.8 $/h.

5.5 Conclusion

Comparing results, total cost from GA is remarkably close to the total cost from the quadprog function. The slight differences were due to power output assignment of each committed unit via quadprog program being done without considering the $i^{th}$ generation unit as reference generation unit. Another thing to point out is that the GA was binary encoded which means it deals with and outputs discrete values compared to values with higher order of decimal points as like the quadprog function. This limits the ability of GA to visit all possible solutions. All in all, it seems that the formulated binary encoded genetic algorithm is working as intended. Advantage of genetic algorithm optimization method is having the ability to expand into more large-scale problems and having the ability to formulate the algorithm in a way to best fit a given optimization problem. In the other hand the particle swarm optimization method was able to achieve a greater global minimum compared to both genetic algorithm and quadratic programming, while also utilizing iterations less than the amount used by genetic algorithm and each iteration was computed at a much faster speed compared to genetic algorithm. Binary genetic algorithm optimization method is a methodology that does not guarantee global minimum, the results of minimum cost are consistent for a given needed iterations on the amount of population but reaches a dead end of minimum result due to the limitation caused by discrete values from the set number of bits. Increasing the number of bits of GA to represent output power for each generating unit would increase computation time even further than it is currently at. This is not an ideal case and could take a long time for the many runs to achieve a more accurate average minimum cost. A better choice of an optimization method would need to have more consistent minimum cost on each run and have the fitness converge at a smaller number of iterations to save time. From the results above, particle swarm optimization meets those requirements which would help develop a better non-linear dynamic economic and environmental dispatch model considering PEVs that would also help propose a suitable dispatch plan for different charging/discharging scenarios.

# Chapter 6

# Dynamic Economic and Environmental Dispatch Considering Electric Vehicles

6.1 Problem setup

The problem setup for PSO DEED Analysis in report will use 10-unit thermal system with operational cost and emission characteristics inspired from [35] and [21] respectively with load demand from [35, 36]. The emission problem data taken from [21] represents 5 units only, thus it will be assumed that the other 5 units of total 10 units will have the same characteristics. Only nitrogen oxides ($NO_x$) are considered in this analysis.

Table (6.1), Table of cost Problem data for the 10-unit system.

| **Unit** | $P_{max}$ **(MW)** | $P_{min}$ **(MW)** | **UR (MW/h)** | **DR (MW/h)** | **a ($/h)** | **b ($/MWh)** | **c ($/MW$^2$h)** |
|---|---|---|---|---|---|---|---|
| **1** | 450 | 150 | 91 | 91 | 670 | 27.79 | 0.00173 |
| **2** | 455 | 150 | 91 | 91 | 970 | 17.26 | 0.00031 |
| **3** | 130 | 20 | 40 | 40 | 700 | 16.60 | 0.002 |
| **4** | 130 | 20 | 50 | 50 | 680 | 16.50 | 0.00211 |
| **5** | 162 | 25 | 50 | 50 | 450 | 19.70 | 0.00211 |
| **6** | 80 | 20 | 20 | 20 | 370 | 22.26 | 0.00712 |
| **7** | 85 | 25 | 20 | 20 | 480 | 27.74 | 0.00079 |
| **8** | 55 | 10 | 11 | 11 | 660 | 25.92 | 0.00413 |
| **9** | 55 | 10 | 11 | 11 | 685 | 27.27 | 0.00222 |
| **10** | 55 | 10 | 11 | 11 | 1000 | 16.19 | 0.00048 |

Table (6.2), Table of emission Problem data for the 10-unit system.

| **Unit** | $\alpha$ **($/h)** | $\beta$ **(lb/MWh)** | $\gamma$ **(lb/MW$^2$h)** | $\eta$ **(lb/h)** | $\delta$ **(1/MW)** |
|---|---|---|---|---|---|
| **1** | 80 | -0.805 | 0.0180 | 0.6550 | 0.02846 |
| **2** | 50 | -0.555 | 0.0150 | 0.5773 | 0.02446 |
| **3** | 60 | -1.355 | 0.0105 | 0.4968 | 0.02270 |
| **4** | 45 | -0.600 | 0.0080 | 0.4860 | 0.01948 |
| **5** | 30 | -0.555 | 0.0120 | 0.5035 | 0.02075 |
| **6** | 80 | -0.805 | 0.0180 | 0.6550 | 0.02846 |
| **7** | 50 | -0.555 | 0.0150 | 0.5773 | 0.02446 |
| **8** | 60 | -1.355 | 0.0105 | 0.4968 | 0.02270 |
| **9** | 45 | -0.600 | 0.0080 | 0.4860 | 0.01948 |
| **10** | 30 | -0.555 | 0.0120 | 0.5035 | 0.02075 |

Table (6.3), containing Particle Swarm Optimization code parameters:

| | |
|---|---|
| Iterations | 300 |
| Particles | 20000 |
| C1 coefficient | 0 |
| C2 coefficient | 1 |
| w – inertia weight maximum | 1 |

Table (6.4), PEVs problem data:

| | |
|---|---|
| Type of PEV | Nissan Altra |
| Number of PEVs | 50000 |
| Capacity | 29.07kWh |
| SoC upon arrival | 20% |
| SoC upon disconnection | 80% |

Calculating total PEV power demand assuming each PEVs charges once a day from [1]:

$$P_{L,ev} = 50000 * 29070 * (80\% - 20\%) = 872.1MW \quad (19)$$

Charging scenarios of plug-in electric vehicles that will be investigated will be five. EPRI profile, off-peak profile, peak profile, stochastic profile that considers the uncertain behavior of drivers when charging, random times during day and night.

6.2 Dispatch analysis without PEVs

Table (6.5), Table of load demand without PEVs.

| **Hour** | **Demand (MW)** | **Hour** | **Demand (MW)** |
|---|---|---|---|
| **1** | 700 | **13** | 1400 |
| **2** | 750 | **13** | 1300 |
| **3** | 850 | **15** | 1200 |
| **4** | 950 | **16** | 1050 |
| **5** | 1000 | **17** | 1000 |
| **6** | 1100 | **18** | 1100 |
| **7** | 1150 | **19** | 1200 |
| **8** | 1200 | **20** | 1400 |
| **9** | 1300 | **21** | 1300 |
| **10** | 1400 | **22** | 1100 |
| **11** | 1450 | **23** | 900 |
| **12** | 1500 | **24** | 800 |

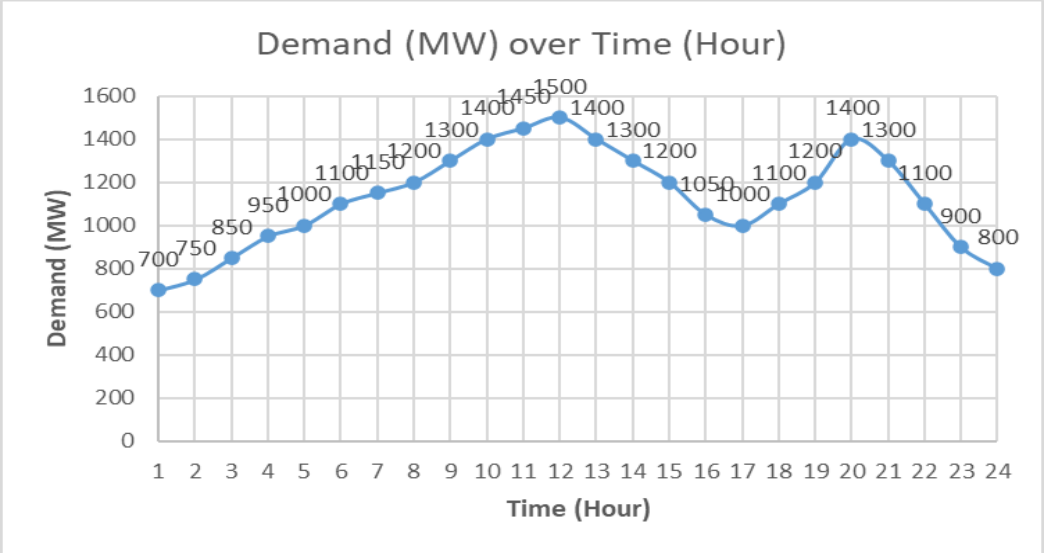


**Fig. 26** Plot of load demand in MW.

Table (6.6), Dynamic Economic Dispatch Analysis Without PEVs:

| | **Total load demand (MW)** | **Total cost ($)** |
|---|---|---|
| **Without PEV** | 27100 | 682160 |

Table (6.7), Dynamic Environmental Dispatch Analysis Without PEVs:

| | **Total load demand (MW)** | **Total emission (lb)** |
|---|---|---|
| **Without PEV** | 27100 | 53210 |

Table (6.8), Dynamic Economic and Environmental Dispatch Analysis Without PEVs:

| | **Total load demand (MW)** | **Total cost ($)** | **Total emission (lb)** |
|---|---|---|---|
| **Without PEV** | 27100 | 702140 | 62984 |

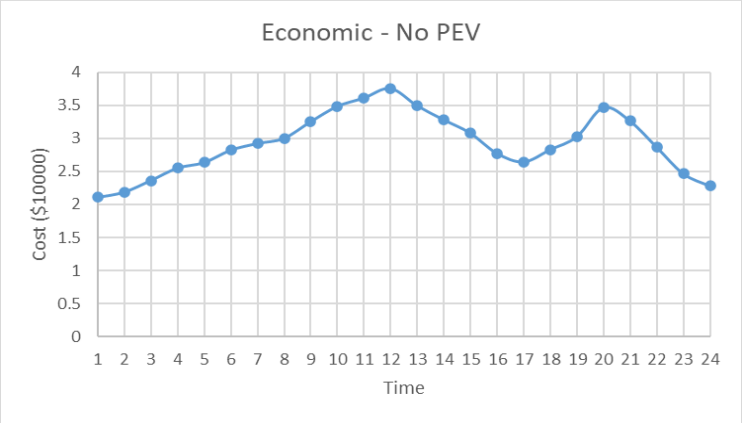


**Fig. 27** Plot of cost over 24 hours for without PEVs.

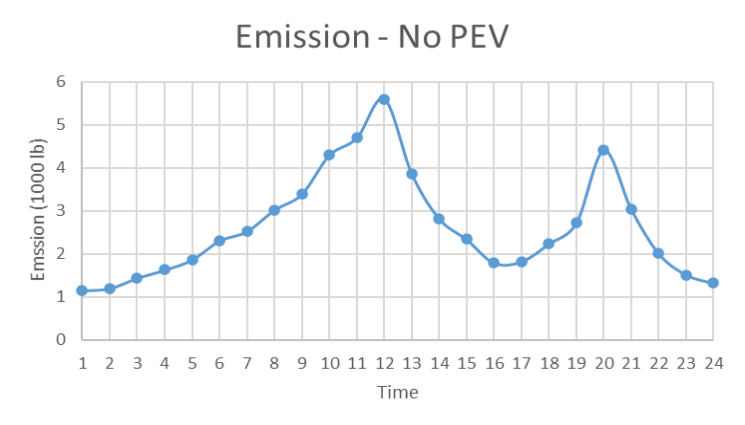


**Fig. 28** Plot of emission over 24 hours for without PEVs.

| GEN 1 (MW) | GEN 2 (MW) | GEN 3 (MW) | GEN 4 (MW) | GEN 5 (MW) | GEN 6 (MW) | GEN 7 (MW) | GEN 8 (MW) | GEN 9 (MW) | GEN 10 (MW) | Total (MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 152.256 | 150.431 | 124.576 | 72.321 | 45.651 | 20.376 | 32.350 | 19.835 | 10.130 | 72.074 | 700 |
| 150.085 | 154.120 | 114.487 | 118.674 | 39.857 | 23.257 | 26.885 | 18.003 | 17.369 | 87.263 | 750 |
| 152.069 | 171.450 | 125.030 | 117.550 | 66.905 | 22.931 | 32.181 | 18.151 | 10.044 | 133.690 | 850 |
| 151.892 | 160.367 | 125.737 | 127.171 | 115.064 | 24.415 | 29.069 | 18.797 | 19.181 | 178.306 | 950 |
| 151.970 | 161.998 | 127.701 | 122.321 | 126.206 | 23.465 | 27.115 | 20.792 | 17.809 | 220.624 | 1000 |
| 153.935 | 175.744 | 129.345 | 126.313 | 157.673 | 30.134 | 33.716 | 23.861 | 11.819 | 257.461 | 1100 |
| 157.775 | 208.134 | 129.352 | 129.909 | 157.489 | 38.931 | 29.705 | 24.502 | 20.559 | 253.643 | 1150 |
| 153.762 | 231.345 | 129.593 | 125.985 | 147.180 | 54.032 | 32.765 | 22.719 | 12.278 | 290.340 | 1200 |
| 183.762 | 231.415 | 129.191 | 129.048 | 159.782 | 66.625 | 48.456 | 33.209 | 21.367 | 297.145 | 1300 |
| 200.938 | 283.605 | 126.306 | 129.440 | 159.202 | 72.974 | 63.002 | 43.991 | 32.179 | 288.363 | 1400 |
| 221.624 | 284.105 | 127.055 | 128.864 | 160.931 | 75.482 | 67.101 | 51.016 | 38.759 | 295.063 | 1450 |
| 255.641 | 284.965 | 123.189 | 128.915 | 161.435 | 76.896 | 81.399 | 43.296 | 48.506 | 295.758 | 1500 |
| 178.019 | 272.143 | 127.544 | 127.064 | 147.667 | 79.036 | 83.247 | 53.907 | 39.505 | 291.866 | 1400 |
| 150.403 | 214.506 | 129.607 | 126.546 | 161.080 | 72.868 | 67.986 | 52.377 | 45.573 | 279.053 | 1300 |
| 153.881 | 170.516 | 126.447 | 126.509 | 159.902 | 70.699 | 54.629 | 43.331 | 34.704 | 259.382 | 1200 |
| 153.166 | 154.674 | 121.090 | 129.760 | 126.385 | 52.343 | 39.661 | 34.198 | 28.401 | 210.322 | 1050 |
| 151.043 | 159.163 | 126.919 | 115.193 | 104.089 | 35.798 | 26.375 | 29.821 | 23.222 | 228.375 | 1000 |
| 150.882 | 177.830 | 129.968 | 118.960 | 138.071 | 41.169 | 28.411 | 38.174 | 16.813 | 259.721 | 1100 |
| 153.400 | 207.554 | 128.520 | 126.602 | 155.450 | 58.970 | 34.653 | 42.177 | 12.345 | 280.330 | 1200 |
| 194.712 | 286.120 | 128.257 | 124.309 | 159.943 | 78.433 | 53.333 | 52.628 | 22.311 | 299.955 | 1400 |
| 153.697 | 233.817 | 128.346 | 127.766 | 159.694 | 67.906 | 59.767 | 53.947 | 31.912 | 283.148 | 1300 |
| 152.539 | 161.956 | 129.454 | 126.525 | 117.168 | 52.191 | 43.007 | 51.676 | 21.419 | 244.064 | 1100 |
| 150.375 | 150.974 | 101.093 | 118.039 | 72.399 | 32.310 | 25.265 | 44.296 | 10.474 | 194.775 | 900 |
| 152.892 | 152.328 | 104.453 | 119.982 | 29.719 | 20.641 | 27.311 | 37.314 | 10.270 | 145.090 | 800 |

Table (6.9), Dynamic Economic and Environmental Dispatch plan without PEVs:

6.2.1 Reviewing results

In order to show the effect of different charging profiles on the load demand, minimum cost of $682160 and a minimum emission of 53210 lb has been obtained via economic dispatch and environmental dispatch as shown in tables (6.6, 6.7), respectively. Considering attaining minimum of both when not excluding the other dispatch using Dynamic economic and environmental dispatch, the total cost and emission becomes $700870 and 64477 lb, respectively as shown in table (6.8). These results will be used to benchmark the cost and emission resulting from different charge profiles.

6.3 Dispatch analysis with peak charging scenario

Table (6.10), Showing the Peak charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 07:00-12:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 13:00-18:00 | 18.5% | 18.5% | 18.5% | 18.5% | 9% | 9% |
| 19:00-24:00 | 4% | 4% | 0% | 0% | 0% | 0% |

Table (6.11), Showing the Peak charging probability distribution power demand for data in Table (4):

| Time | PEVs Power Demand (MW) | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 07:00-12:00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13:00-18:00 | 161.340 | 161.340 | 161.340 | 161.34 | 78.489 | 78.489 |
| 19:00-24:00 | 34.884 | 34.884 | 0 | 0 | 0 | 0 |

Table (6.12), Table of load demand with PEVs (Peak charge profile).

| **Hour** | **Demand (MW)** | **Hour** | **Demand (MW)** |
|---|---|---|---|
| **1** | 700 | **13** | 1561.340 |
| **2** | 750 | **13** | 1461.340 |
| **3** | 850 | **15** | 1361.340 |
| **4** | 950 | **16** | 1211.340 |
| **5** | 1000 | **17** | 1078.489 |
| **6** | 1100 | **18** | 1178.489 |
| **7** | 1150 | **19** | 1234.884 |
| **8** | 1200 | **20** | 1434.884 |
| **9** | 1300 | **21** | 1300 |
| **10** | 1400 | **22** | 1100 |
| **11** | 1450 | **23** | 900 |
| **12** | 1500 | **24** | 800 |

The plug-in electric vehicles' charging total daily load demand ($872.1MW$) is met through percentages from the total inside hours between (13:00) and (20:00) as shown in table (6.10).

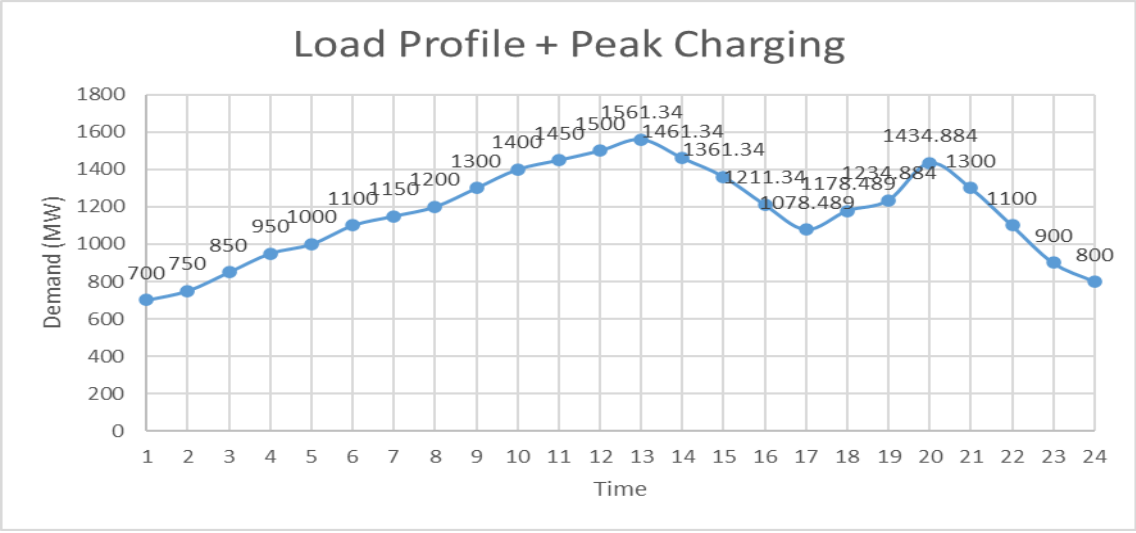


**Fig. 29** Plot of load profile considering Peak charging.

Table (6.13), Dynamic Economic Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) |
|---|---|---|
| **Peak Charging** | 27972 | 699850 |

Table (6.14), Dynamic Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total emission (lb) |
|---|---|---|
| **Peak Charging** | 27972 | 62427 |

Table (6.15), Dynamic Economic and Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) | Total emission (lb) |
|---|---|---|---|
| **Peak Charging** | 27972 | 720980 | 71903 |

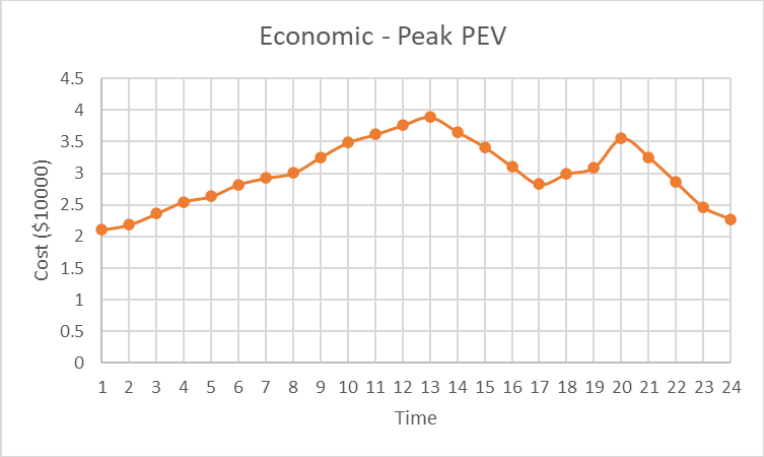


**Fig. 30** Plot of cost over 24 hours for Peak charging PEVs.

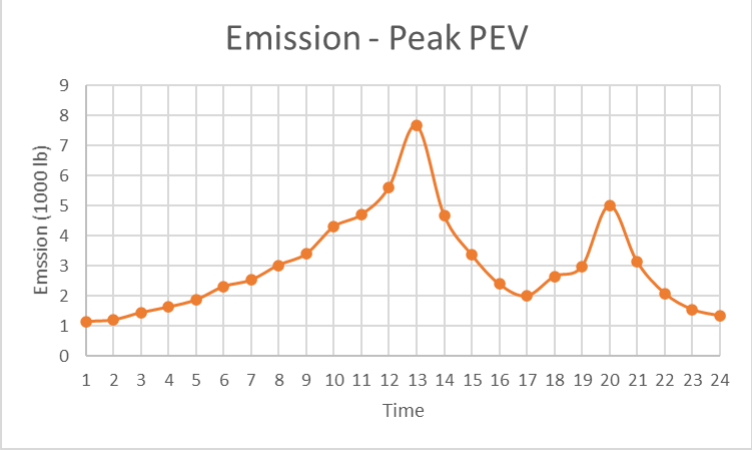


**Fig. 31** Plot of emission over 24 hours for Peak charging PEVs.

Table (6.16), Dynamic Economic and Environmental Dispatch plan with Peak scenario:

| GEN 1 (MW) | GEN 2 (MW) | GEN 3 (MW) | GEN 4 (MW) | GEN 5 (MW) | GEN 6 (MW) | GEN 7 (MW) | GEN 8 (MW) | GEN 9 (MW) | GEN 10 (MW) | Total (MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 152.256 | 150.431 | 124.576 | 72.321 | 45.651 | 20.376 | 32.350 | 19.835 | 10.130 | 72.074 | 700 |
| 150.085 | 154.120 | 114.487 | 118.674 | 39.857 | 23.257 | 26.885 | 18.003 | 17.369 | 87.263 | 750 |
| 152.069 | 171.450 | 125.030 | 117.550 | 66.905 | 22.931 | 32.181 | 18.151 | 10.044 | 133.690 | 850 |
| 151.892 | 160.367 | 125.737 | 127.171 | 115.064 | 24.415 | 29.069 | 18.797 | 19.181 | 178.306 | 950 |
| 151.970 | 161.998 | 127.701 | 122.321 | 126.206 | 23.465 | 27.115 | 20.792 | 17.809 | 220.624 | 1000 |
| 153.935 | 175.744 | 129.345 | 126.313 | 157.673 | 30.134 | 33.716 | 23.861 | 11.819 | 257.461 | 1100 |
| 157.775 | 208.134 | 129.352 | 129.909 | 157.489 | 38.931 | 29.705 | 24.502 | 20.559 | 253.643 | 1150 |
| 153.762 | 231.345 | 129.593 | 125.985 | 147.180 | 54.032 | 32.765 | 22.719 | 12.278 | 290.340 | 1200 |
| 183.762 | 231.415 | 129.191 | 129.048 | 159.782 | 66.625 | 48.456 | 33.209 | 21.367 | 297.145 | 1300 |
| 200.938 | 283.605 | 126.306 | 129.440 | 159.202 | 72.974 | 63.002 | 43.991 | 32.179 | 288.363 | 1400 |
| 221.624 | 284.105 | 127.055 | 128.864 | 160.931 | 75.482 | 67.101 | 51.016 | 38.759 | 295.063 | 1450 |
| 255.641 | 284.965 | 123.189 | 128.915 | 161.435 | 76.896 | 81.399 | 43.296 | 48.506 | 295.758 | 1500 |
| 267.215 | 329.673 | 124.012 | 128.262 | 155.988 | 78.284 | 83.228 | 51.261 | 48.028 | 295.391 | 1561.34 |
| 216.828 | 287.774 | 120.291 | 129.889 | 151.817 | 79.995 | 79.792 | 50.025 | 50.865 | 294.063 | 1461.34 |
| 155.502 | 255.488 | 129.693 | 122.779 | 160.431 | 77.871 | 79.375 | 43.259 | 54.692 | 282.249 | 1361.34 |
| 150.938 | 188.277 | 126.656 | 126.441 | 154.349 | 69.018 | 61.146 | 32.666 | 46.161 | 255.689 | 1211.34 |
| 152.137 | 160.410 | 117.584 | 122.511 | 129.413 | 50.677 | 42.537 | 25.476 | 36.880 | 240.865 | 1078.489 |
| 151.731 | 198.035 | 124.315 | 128.773 | 136.576 | 55.538 | 37.012 | 30.613 | 29.268 | 286.628 | 1178.489 |
| 152.351 | 232.892 | 129.680 | 125.200 | 153.569 | 63.666 | 27.589 | 37.532 | 29.635 | 282.771 | 1234.884 |
| 216.870 | 299.289 | 128.891 | 129.828 | 161.432 | 77.325 | 39.356 | 47.024 | 39.304 | 295.566 | 1434.884 |
| 155.399 | 234.702 | 128.280 | 128.969 | 159.669 | 63.536 | 50.509 | 52.326 | 33.968 | 292.642 | 1300 |
| 151.398 | 176.756 | 126.163 | 126.917 | 112.191 | 44.168 | 34.786 | 50.565 | 28.118 | 248.938 | 1100 |
| 150.371 | 151.114 | 98.747 | 118.126 | 67.696 | 24.288 | 25.208 | 43.465 | 17.173 | 203.812 | 900 |
| 150.007 | 152.376 | 128.714 | 80.954 | 25.242 | 29.149 | 26.700 | 36.968 | 11.725 | 158.164 | 800 |

### 6.3.1 Reviewing results

The results considering the peak charging profile over a 24-hour period, show a minimum cost of $699850 and a minimum emission of 62427 lb has been obtained via economic dispatch and environmental dispatch as shown in tables (6.13, 6.14), respectively. Considering attaining minimum of both when not excluding the other dispatch using Dynamic economic and environmental dispatch, the total cost and emission becomes $720980 and 71903 lb, respectively as shown in table (6.15).

6.4 Dispatch analysis with off-peak charging scenario

Table (6.17), Showing the Off-peak charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 18.5% | 18.5% | 9% | 9% | 4% | 4% |
| 07:00-12:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 13:00-18:00 | 0% | 0% | 0% | 0% | 0% | 0% |
| 19:00-24:00 | 0% | 0% | 0% | 0% | 18.5% | 18.5% |

Table (6.18), Showing the Off-Peak charging probability distribution power demand for data in Table (4):

| Time | PEVs Power Demand (MW) | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 161.340 | 161.340 | 78.489 | 78.489 | 34.884 | 34.884 |
| 07:00-12:00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13:00-18:00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19:00-24:00 | 0 | 0 | 0 | 0 | 161.340 | 161.340 |

Table (6.19), Table of load demand with PEVs (Off-Peak charge profile).

| **Hour** | **Demand (MW)** | **Hour** | **Demand (MW)** |
|---|---|---|---|
| **1** | 861.340 | **13** | 1400 |
| **2** | 911.340 | **13** | 1300 |
| **3** | 928.489 | **15** | 1200 |
| **4** | 1028.489 | **16** | 1050 |
| **5** | 1034.884 | **17** | 1000 |
| **6** | 1134.884 | **18** | 1100 |
| **7** | 1150 | **19** | 1200 |
| **8** | 1200 | **20** | 1400 |
| **9** | 1300 | **21** | 1300 |
| **10** | 1400 | **22** | 1100 |
| **11** | 1450 | **23** | 1061.340 |
| **12** | 1500 | **24** | 961.340 |

The plug-in electric vehicles' charging total daily load demand ($872.1MW$) is met through percentages from the total inside hours between (23:00) and (06:00) according to table (6.17).

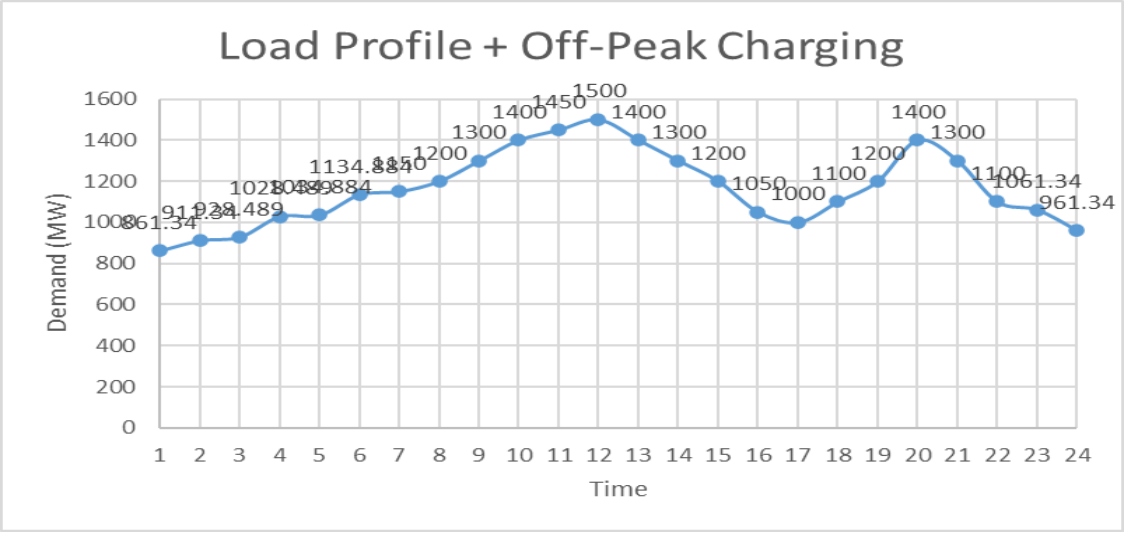


**Fig. 32** Plot of load profile considering Off-Peak

Table (6.20), Dynamic Economic Dispatch Analysis with PEVs:

| | **Total load demand (MW)** | **Total cost ($)** |
|---|---|---|
| **Off Peak Charging** | 27972 | 696710 |

Table (6.21), Dynamic Environmental Dispatch Analysis with PEVs:

| | **Total load demand (MW)** | **Total emission (lb)** |
|---|---|---|
| **Off Peak Charging** | 27972 | 55146 |

Table (6.22), Dynamic Economic and Environmental Dispatch Analysis with PEVs:

| | **Total load demand (MW)** | **Total cost ($)** | **Total emission (lb)** |
|---|---|---|---|
| **Off Peak Charging** | 27972 | 716560 | 67922 |

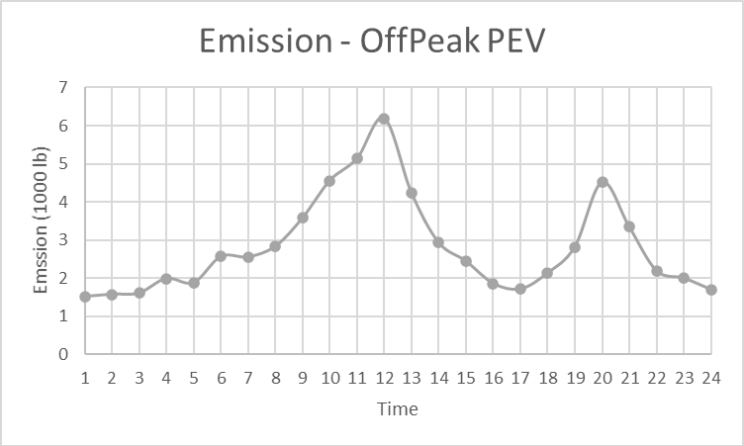


**Fig. 33** Plot of cost over 24 hours for off-Peak charging PEVs.

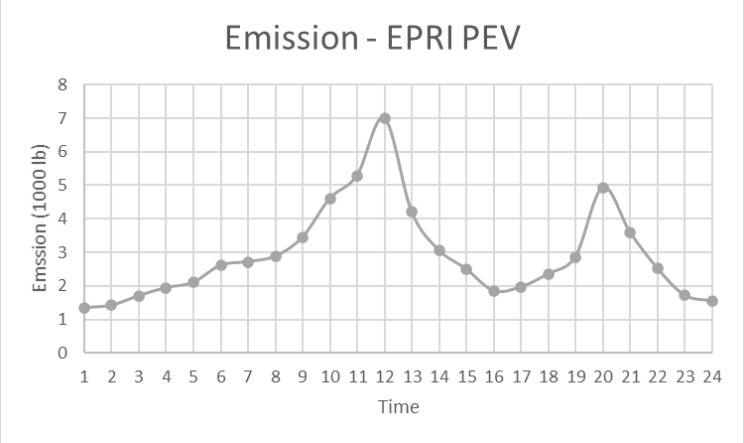


**Fig. 34** Plot of emission over 24 hours for off-Peak charging PEVs.

Table (6.23), Dynamic Economic and Environmental Dispatch plan with Off-Peak scenario:

| GEN 1 (MW) | GEN 2 (MW) | GEN 3 (MW) | GEN 4 (MW) | GEN 5 (MW) | GEN 6 (MW) | GEN 7 (MW) | GEN 8 (MW) | GEN 9 (MW) | GEN 10 (MW) | Total (MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 161.182 | 170.652 | 125.142 | 119.758 | 54.400 | 27.316 | 25.281 | 13.895 | 17.663 | 146.051 | 861.34 |
| 151.481 | 165.374 | 128.232 | 117.695 | 76.244 | 33.426 | 28.445 | 19.848 | 10.562 | 180.032 | 911.34 |
| 150.849 | 163.610 | 121.454 | 116.648 | 87.944 | 50.730 | 25.129 | 14.153 | 12.670 | 185.302 | 928.489 |
| 150.439 | 179.924 | 129.655 | 126.183 | 84.205 | 63.193 | 25.089 | 16.874 | 18.797 | 234.128 | 1028.489 |
| 151.668 | 165.217 | 128.860 | 126.068 | 124.585 | 59.659 | 31.141 | 18.394 | 13.979 | 215.314 | 1034.884 |
| 154.014 | 226.017 | 127.415 | 129.330 | 141.934 | 51.349 | 25.913 | 18.570 | 13.835 | 246.506 | 1134.884 |
| 151.839 | 220.454 | 129.623 | 128.710 | 138.864 | 56.657 | 33.808 | 17.143 | 19.158 | 253.744 | 1150 |
| 154.574 | 215.315 | 119.676 | 129.179 | 155.880 | 72.259 | 37.929 | 18.309 | 15.619 | 281.260 | 1200 |
| 152.242 | 263.283 | 115.762 | 128.380 | 160.484 | 79.183 | 50.690 | 26.615 | 26.370 | 296.991 | 1300 |
| 192.038 | 293.630 | 125.933 | 128.069 | 160.359 | 75.407 | 68.661 | 33.977 | 24.748 | 297.177 | 1400 |
| 228.019 | 296.050 | 129.465 | 126.495 | 157.033 | 73.154 | 82.685 | 31.420 | 30.286 | 295.392 | 1450 |
| 257.676 | 303.564 | 128.800 | 128.565 | 159.204 | 70.131 | 81.630 | 34.589 | 38.523 | 297.319 | 1500 |
| 170.626 | 289.270 | 117.808 | 129.849 | 161.488 | 73.317 | 75.370 | 44.070 | 42.279 | 295.924 | 1400 |
| 151.052 | 224.220 | 125.518 | 129.364 | 154.260 | 75.664 | 60.893 | 48.021 | 46.760 | 284.246 | 1300 |
| 151.177 | 207.266 | 128.639 | 127.150 | 139.314 | 56.677 | 45.264 | 39.765 | 48.428 | 256.319 | 1200 |
| 150.439 | 151.754 | 113.853 | 124.737 | 141.444 | 48.001 | 27.316 | 29.258 | 37.960 | 225.238 | 1050 |
| 150.605 | 170.067 | 127.777 | 123.291 | 134.115 | 41.201 | 25.177 | 21.001 | 28.341 | 178.425 | 1000 |
| 151.650 | 195.020 | 121.984 | 126.712 | 143.103 | 51.066 | 26.829 | 27.961 | 31.269 | 224.406 | 1100 |
| 150.365 | 227.474 | 122.399 | 125.054 | 159.624 | 57.616 | 30.800 | 21.096 | 32.243 | 273.328 | 1200 |
| 210.185 | 284.345 | 125.692 | 129.842 | 158.386 | 76.331 | 46.582 | 29.869 | 42.346 | 296.420 | 1400 |
| 150.737 | 249.894 | 128.606 | 129.069 | 152.713 | 64.539 | 49.518 | 32.177 | 44.702 | 298.044 | 1300 |
| 153.002 | 173.933 | 119.166 | 126.929 | 135.970 | 47.047 | 29.621 | 21.672 | 37.518 | 255.143 | 1100 |
| 151.500 | 150.653 | 125.630 | 129.102 | 161.560 | 30.643 | 28.759 | 19.337 | 30.949 | 233.207 | 1061.34 |
| 151.725 | 154.524 | 129.794 | 112.294 | 115.805 | 31.609 | 27.640 | 14.092 | 20.571 | 203.285 | 961.34 |

6.4.1 Reviewing results

The results considering the off-peak charging profile over a 24-hour period, show a minimum cost of $696710 and a minimum emission of 55146 lb has been obtained via economic dispatch and environmental dispatch as shown in tables (6.20, 6.21), respectively. Considering attaining minimum of both when not excluding the other dispatch using Dynamic economic and environmental dispatch, the total cost and emission becomes $716560 and 67922 lb, respectively as shown in table (22).

6.5 Dispatch analysis with EPRI charging scenario

Table (6.24), Showing the EPRI charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 10% | 10% | 9.5% | 7% | 5% | 3% |
| 07:00-12:00 | 1% | 0.3% | 0.3% | 1.3% | 2.1% | 2.1% |
| 13:00-18:00 | 2.1% | 2.1% | 2.1% | 1% | 0.5% | 0.5% |
| 19:00-24:00 | 1.6% | 3.6% | 5.4% | 9.5% | 10% | 10% |

Table (6.25), Showing the EPRI charging probability distribution power demand for data in Table (4):

| Time | PEVs Power Demand (MW) | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 87.21 | 87.21 | 82.85 | 61.05 | 43.61 | 26.16 |
| 07:00-12:00 | 8.72 | 2.62 | 2.62 | 11.34 | 18.31 | 18.31 |
| 13:00-18:00 | 18.31 | 18.31 | 18.31 | 8.72 | 4.36 | 4.36 |
| 19:00-24:00 | 13.95 | 31.4 | 47.09 | 82.85 | 87.21 | 87.21 |

Table (6.26), Table of load demand with PEVs (EPRI charge profile).

| **Hour** | **Demand (MW)** | **Hour** | **Demand (MW)** |
|---|---|---|---|
| **1** | 787.21 | **13** | 1418.31 |
| **2** | 837.21 | **13** | 1318.31 |
| **3** | 932.85 | **15** | 1218.31 |
| **4** | 1011.05 | **16** | 1058.72 |
| **5** | 1043.61 | **17** | 1004.36 |
| **6** | 1126.16 | **18** | 1104.36 |
| **7** | 1158.72 | **19** | 1213.95 |
| **8** | 1202.62 | **20** | 1431.4 |
| **9** | 1302.62 | **21** | 1347.09 |
| **10** | 1411.34 | **22** | 1182.85 |
| **11** | 1468.31 | **23** | 987.21 |
| **12** | 1518.31 | **24** | 887.21 |

The plug-in electric vehicles' charging total daily load demand ($872.1MW$) is met through random different percentages from the total which is taking up the whole day (24 – hours) according to table (6.24).

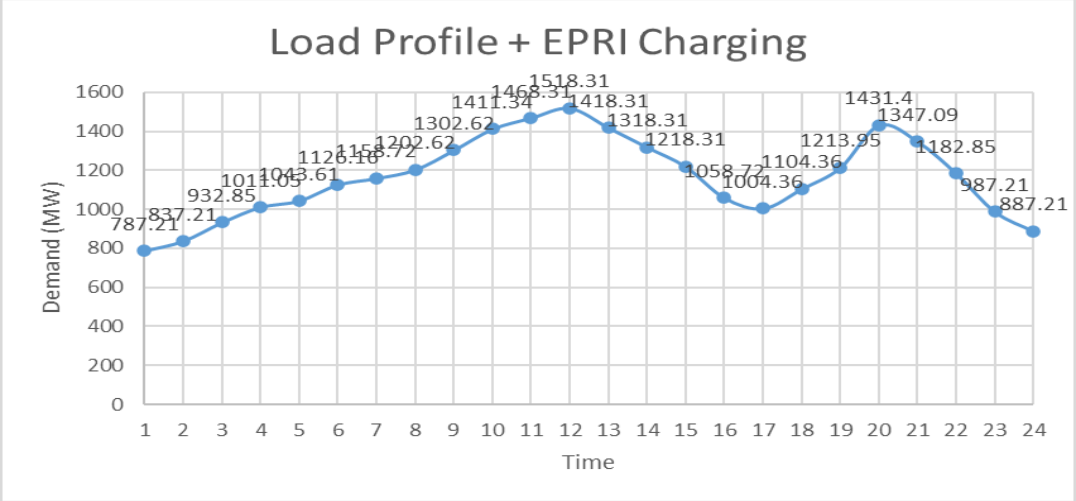


**Fig. 35** Plot of load profile considering EPRI

Table (6.27), Dynamic Economic Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) |
|---|---|---|
| **EPRI Charging** | 27972 | 697810 |

Table (6.28), Dynamic Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total emission (lb) |
|---|---|---|
| **EPRI Charging** | 27972 | 57169 |

Table (6.29), Dynamic Economic and Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) | Total emission (lb) |
|---|---|---|---|
| **EPRI Charging** | 27972 | 717020 | 70177 |

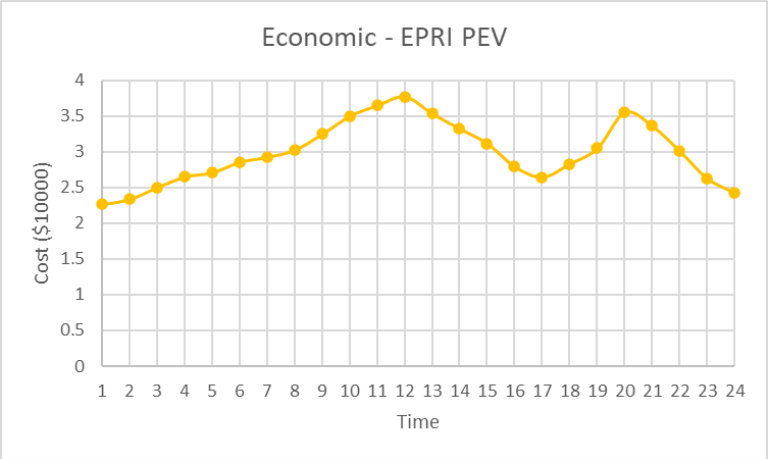


**Fig. 36** Plot of cost over 24 hours for EPRI charging PEVs.

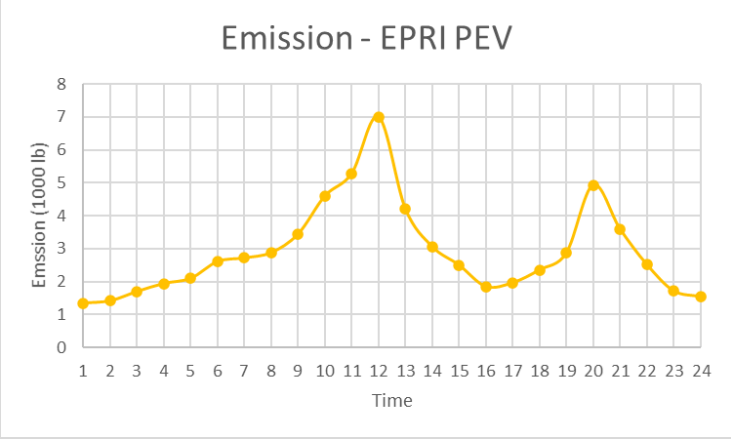


**Fig. 37** Plot of emission over 24 hours for EPRI charging PEVs.

Table (6.30), Dynamic Economic and Environmental Dispatch plan with EPRI scenario:

| GEN 1 (MW) | GEN 2 (MW) | GEN 3 (MW) | GEN 4 (MW) | GEN 5 (MW) | GEN 6 (MW) | GEN 7 (MW) | GEN 8 (MW) | GEN 9 (MW) | GEN 10 (MW) | Total (MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 155.546 | 151.614 | 83.429 | 126.655 | 35.874 | 21.274 | 47.458 | 11.782 | 13.878 | 139.699 | 787.21 |
| 150.485 | 152.468 | 116.087 | 125.333 | 25.777 | 38.047 | 32.381 | 20.262 | 10.287 | 166.084 | 837.21 |
| 150.684 | 178.769 | 126.309 | 125.851 | 75.651 | 29.600 | 27.991 | 15.892 | 13.021 | 189.083 | 932.85 |
| 152.482 | 186.580 | 128.550 | 127.395 | 119.150 | 35.144 | 27.179 | 14.662 | 11.585 | 208.322 | 1011.05 |
| 151.161 | 180.720 | 128.592 | 120.276 | 139.883 | 23.572 | 26.426 | 24.259 | 11.629 | 237.092 | 1043.61 |
| 150.745 | 202.480 | 123.875 | 127.612 | 154.196 | 26.671 | 26.649 | 19.362 | 17.568 | 277.002 | 1126.16 |
| 154.575 | 214.015 | 127.996 | 128.277 | 154.680 | 43.842 | 26.539 | 13.664 | 21.220 | 273.910 | 1158.72 |
| 151.329 | 212.751 | 125.730 | 126.335 | 149.059 | 62.576 | 39.998 | 14.194 | 27.782 | 292.866 | 1202.62 |
| 162.891 | 250.669 | 129.061 | 124.352 | 157.367 | 77.500 | 58.229 | 13.456 | 35.064 | 294.030 | 1302.62 |
| 195.934 | 292.563 | 129.357 | 129.811 | 158.521 | 68.285 | 75.691 | 20.404 | 41.855 | 298.920 | 1411.34 |
| 225.403 | 300.734 | 128.844 | 129.866 | 160.128 | 77.774 | 82.723 | 17.508 | 47.632 | 297.697 | 1468.31 |
| 251.438 | 327.739 | 121.355 | 127.862 | 157.520 | 77.881 | 83.272 | 19.401 | 52.859 | 298.983 | 1518.31 |
| 184.064 | 281.643 | 122.396 | 129.694 | 160.592 | 78.446 | 81.353 | 29.046 | 54.772 | 296.306 | 1418.31 |
| 154.655 | 227.475 | 126.616 | 128.812 | 156.909 | 70.824 | 84.383 | 32.557 | 49.384 | 286.696 | 1318.31 |
| 151.876 | 206.543 | 126.790 | 122.147 | 154.650 | 59.736 | 68.198 | 35.925 | 38.453 | 253.992 | 1218.31 |
| 150.238 | 167.995 | 124.209 | 122.941 | 127.790 | 49.880 | 50.645 | 25.106 | 29.747 | 210.168 | 1058.72 |
| 150.331 | 211.823 | 126.098 | 126.475 | 92.863 | 36.321 | 32.869 | 17.659 | 19.001 | 190.921 | 1004.36 |
| 150.156 | 213.932 | 116.935 | 127.657 | 142.829 | 47.131 | 26.548 | 26.904 | 13.460 | 238.807 | 1104.36 |
| 151.497 | 226.011 | 129.666 | 129.605 | 149.035 | 54.992 | 32.565 | 35.514 | 23.109 | 281.957 | 1213.95 |
| 223.989 | 291.589 | 129.631 | 128.975 | 157.915 | 73.878 | 47.577 | 46.470 | 31.931 | 299.444 | 1431.4 |
| 174.589 | 254.205 | 129.819 | 126.113 | 155.071 | 68.682 | 62.171 | 47.624 | 29.857 | 298.962 | 1347.09 |
| 150.368 | 210.821 | 128.936 | 127.226 | 146.668 | 52.366 | 47.065 | 37.714 | 21.601 | 260.086 | 1182.85 |
| 152.867 | 152.366 | 127.573 | 123.637 | 101.949 | 41.503 | 31.225 | 33.569 | 11.394 | 211.127 | 987.21 |
| 150.477 | 165.878 | 119.510 | 120.712 | 54.155 | 27.541 | 27.440 | 28.737 | 10.661 | 182.098 | 887.21 |

6.5.1 Reviewing results

The results considering the EPRI charging profile over a 24-hour period, show a minimum cost of $697810 and a minimum emission of 57169 lb has been obtained via economic dispatch and environmental dispatch as shown in tables (6.27, 6.28), respectively. Considering attaining minimum of both when not excluding the other dispatch using Dynamic economic and environmental dispatch, the total cost and emission becomes $717020 and 70177 lb, respectively as shown in table (6.29).

6.6 Dispatch analysis with stochastic charging scenario

Table (6.31), Showing the Stochastic charging probability distribution [1]:

| Time | Probability distribution Percentage | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 5.7% | 4.9% | 4.8% | 2.4% | 2.6% | 9.7% |
| 07:00-12:00 | 8.7% | 4.8% | 1.1% | 3.2% | 2.1% | 5.7% |
| 13:00-18:00 | 3.8% | 2.2% | 2.1% | 6.1% | 3.2% | 2.2% |
| 19:00-24:00 | 2.8% | 2.2% | 5.5% | 2.5% | 3.5% | 8.2% |

Table (6.32), Showing the Stochastic charging probability distribution power demand for data in Table (4):

| Time | PEVs Power Demand (MW) | | | | | |
|---|---|---|---|---|---|---|
| 01:00-06:00 | 49.71 | 42.73 | 41.86 | 20.93 | 22.67 | 84.59 |
| 07:00-12:00 | 75.87 | 41.86 | 9.59 | 27.91 | 18.31 | 49.71 |
| 13:00-18:00 | 33.14 | 19.19 | 18.31 | 53.2 | 27.91 | 19.19 |
| 19:00-24:00 | 24.42 | 19.19 | 47.97 | 21.8 | 30.52 | 71.51 |

Table (6.33), Table of load demand with PEVs (Stochastic charge profile).

| **Hour** | **Demand (MW)** | **Hour** | **Demand (MW)** |
|---|---|---|---|
| **1** | 749.71 | **13** | 1433.14 |
| **2** | 792.73 | **13** | 1319.19 |
| **3** | 891.86 | **15** | 1218.31 |
| **4** | 970.93 | **16** | 1103.2 |
| **5** | 1022.67 | **17** | 1027.91 |
| **6** | 1184.59 | **18** | 1119.19 |
| **7** | 1225.87 | **19** | 1224.42 |
| **8** | 1241.86 | **20** | 1419.19 |
| **9** | 1309.59 | **21** | 1347.97 |
| **10** | 1427.91 | **22** | 1121.8 |
| **11** | 1468.31 | **23** | 930.52 |
| **12** | 1549.71 | **24** | 871.51 |

The plug-in electric vehicles' charging total daily load demand ($872.1MW$) is met through different percentages from the total which is taking up the whole day (24 – hours) according to table (6.31).

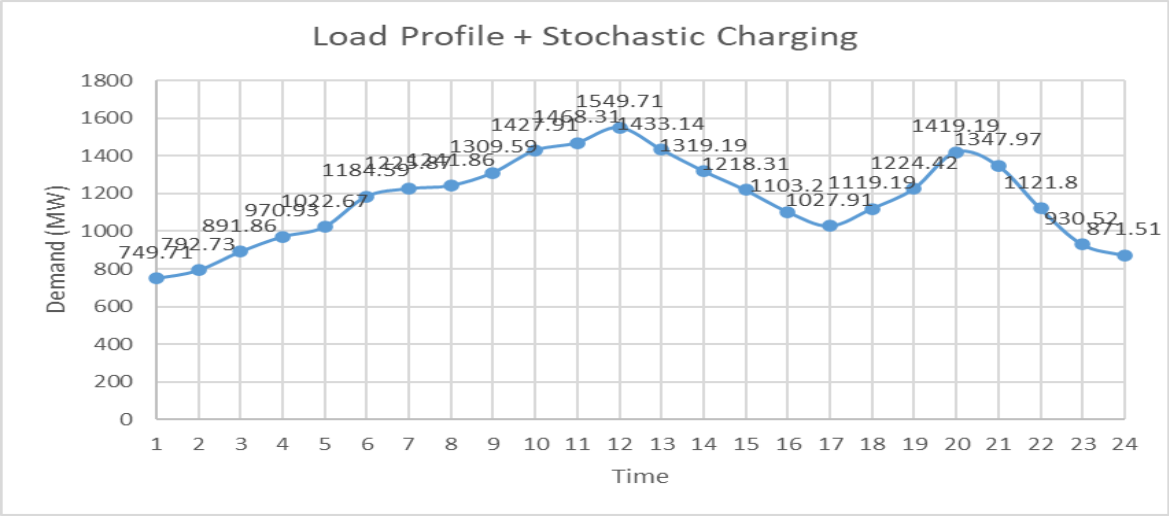


**Fig. 38** Plot of load profile considering stochastic

Table (6.34), Dynamic Economic Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) |
|---|---|---|
| **Stochastic Charging** | 27972 | 699290 |

Table (6.35), Dynamic Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total emission (lb) |
|---|---|---|
| **Stochastic Charging** | 27972 | 58782 |

Table (6.36), Dynamic Economic and Environmental Dispatch Analysis with PEVs:

| | Total load demand (MW) | Total cost ($) | Total emission (lb) |
|---|---|---|---|
| **Stochastic Charging** | 27972 | 717390 | 71495 |

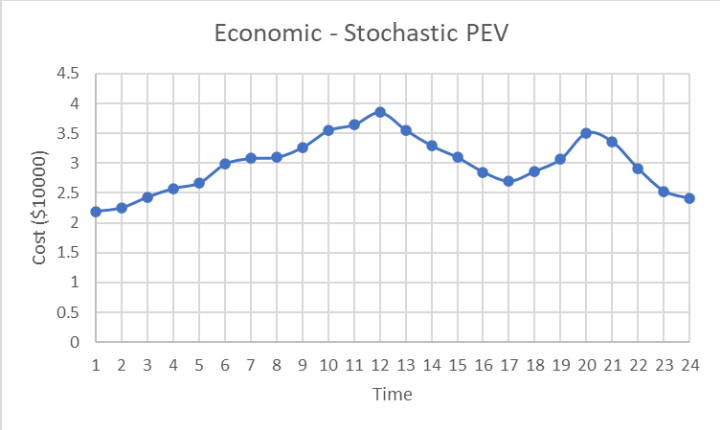


**Fig. 39** Plot of cost over 24 hours for Stochastic charging PEVs.

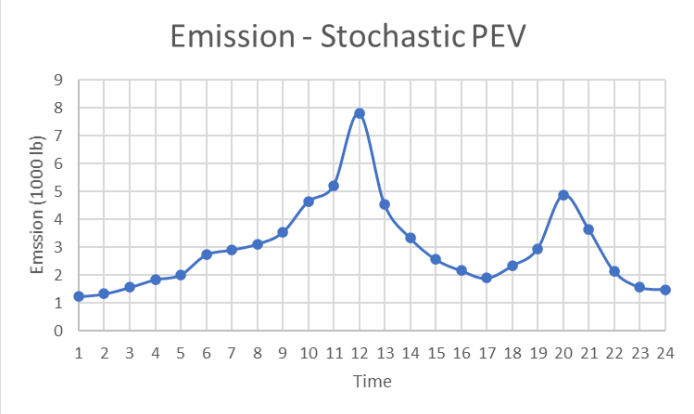


**Fig. 40** Plot of emission over 24 hours for Stochastic charging PEVs.

Table (6.37), Dynamic Economic and Environmental Dispatch plan with Stochastic scenario:

| GEN 1 (MW) | GEN 2 (MW) | GEN 3 (MW) | GEN 4 (MW) | GEN 5 (MW) | GEN 6 (MW) | GEN 7 (MW) | GEN 8 (MW) | GEN 9 (MW) | GEN 10 (MW) | Total (MW) |
|---|---|---|---|---|---|---|---|---|---|---|
| 152.256 | 150.431 | 124.576 | 72.321 | 45.651 | 20.376 | 32.350 | 19.835 | 10.130 | 121.784 | 749.71 |
| 151.524 | 154.495 | 117.446 | 112.968 | 33.156 | 33.692 | 28.869 | 11.418 | 10.097 | 139.065 | 792.73 |
| 152.012 | 168.752 | 128.071 | 123.706 | 61.450 | 33.756 | 30.604 | 10.322 | 12.988 | 170.199 | 891.86 |
| 152.357 | 189.710 | 124.087 | 128.835 | 68.887 | 48.793 | 26.052 | 18.724 | 11.136 | 202.349 | 970.93 |
| 150.874 | 182.229 | 124.730 | 126.849 | 110.948 | 44.453 | 28.181 | 10.854 | 13.260 | 230.292 | 1022.67 |
| 152.279 | 210.097 | 127.411 | 129.525 | 153.063 | 64.182 | 38.579 | 18.908 | 12.785 | 277.762 | 1184.59 |
| 152.126 | 240.047 | 125.189 | 129.203 | 158.881 | 77.808 | 41.202 | 28.837 | 12.022 | 260.556 | 1225.87 |
| 151.947 | 226.828 | 129.582 | 127.438 | 150.080 | 77.190 | 34.764 | 25.575 | 18.715 | 299.740 | 1241.86 |
| 169.688 | 253.508 | 125.389 | 122.491 | 161.737 | 71.928 | 52.131 | 32.760 | 24.216 | 295.744 | 1309.59 |
| 205.965 | 290.001 | 125.753 | 126.965 | 156.298 | 76.886 | 70.308 | 42.819 | 33.592 | 299.322 | 1427.91 |
| 219.894 | 304.068 | 128.837 | 126.098 | 161.356 | 74.398 | 77.601 | 53.431 | 29.368 | 293.259 | 1468.31 |
| 262.004 | 338.259 | 129.779 | 126.995 | 160.256 | 78.130 | 83.424 | 53.833 | 36.733 | 280.297 | 1549.71 |
| 188.652 | 294.873 | 127.705 | 128.856 | 153.171 | 79.625 | 69.452 | 53.912 | 39.616 | 297.277 | 1433.14 |
| 150.516 | 260.971 | 129.990 | 126.322 | 153.835 | 78.540 | 55.697 | 51.762 | 30.790 | 280.766 | 1319.19 |
| 152.947 | 218.178 | 127.600 | 129.238 | 147.709 | 65.985 | 45.430 | 46.936 | 29.684 | 254.602 | 1218.31 |
| 151.761 | 160.764 | 116.609 | 129.618 | 158.786 | 50.598 | 25.657 | 36.066 | 19.872 | 253.468 | 1103.2 |
| 152.226 | 164.879 | 126.192 | 123.735 | 146.904 | 32.067 | 27.351 | 25.698 | 15.179 | 213.681 | 1027.91 |
| 155.032 | 177.197 | 127.675 | 128.155 | 159.235 | 42.610 | 32.368 | 26.660 | 11.485 | 258.774 | 1119.19 |
| 157.471 | 212.393 | 127.436 | 128.421 | 158.815 | 61.124 | 37.502 | 34.889 | 11.612 | 294.757 | 1224.42 |
| 204.518 | 299.498 | 128.642 | 127.969 | 158.110 | 79.788 | 55.183 | 45.285 | 21.119 | 299.078 | 1419.19 |
| 161.553 | 267.131 | 124.900 | 126.299 | 159.792 | 68.147 | 73.080 | 54.762 | 19.648 | 292.658 | 1347.97 |
| 151.060 | 182.266 | 125.457 | 127.214 | 122.671 | 50.424 | 55.696 | 51.187 | 10.021 | 245.804 | 1121.8 |
| 150.342 | 150.711 | 120.690 | 100.182 | 82.319 | 31.706 | 35.937 | 42.507 | 11.317 | 204.809 | 930.52 |
| 150.413 | 173.470 | 122.175 | 121.618 | 38.026 | 26.560 | 28.186 | 35.258 | 14.945 | 160.857 | 871.51 |

6.6.1 Reviewing results

The results considering the stochastic charging profile over a 24-hour period, show a minimum cost of $699290 and a minimum emission of 58782 lb has been obtained via economic dispatch and environmental dispatch as shown in tables (6.34, 6.35), respectively. Considering attaining minimum of both when not excluding the other dispatch using Dynamic economic and environmental dispatch, the total cost and emission becomes $717390 and 71495 lb, respectively as shown in table (6.36).

6.7 Conclusion

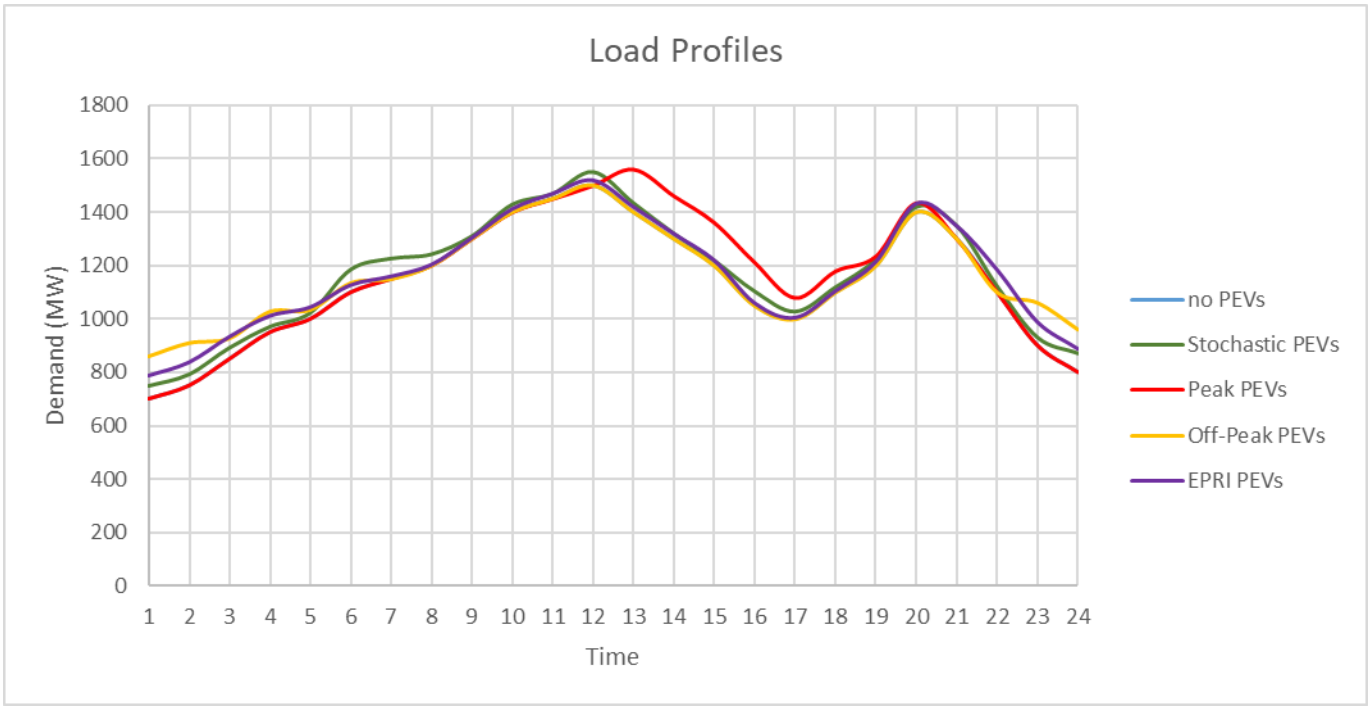


**Fig. 41** Plot of load profiles considering PEVs.

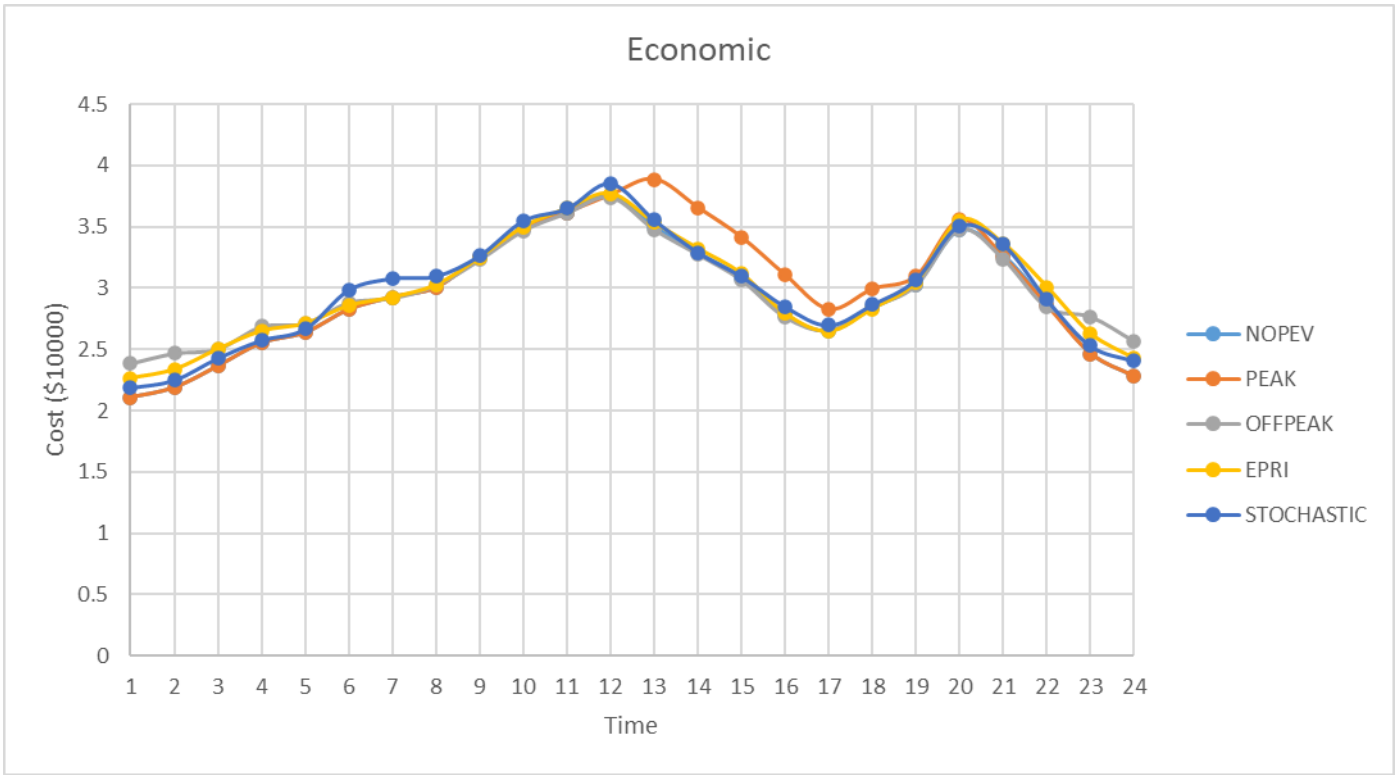


**Fig. 42** Plot of cost over 24 hours considering PEVs.

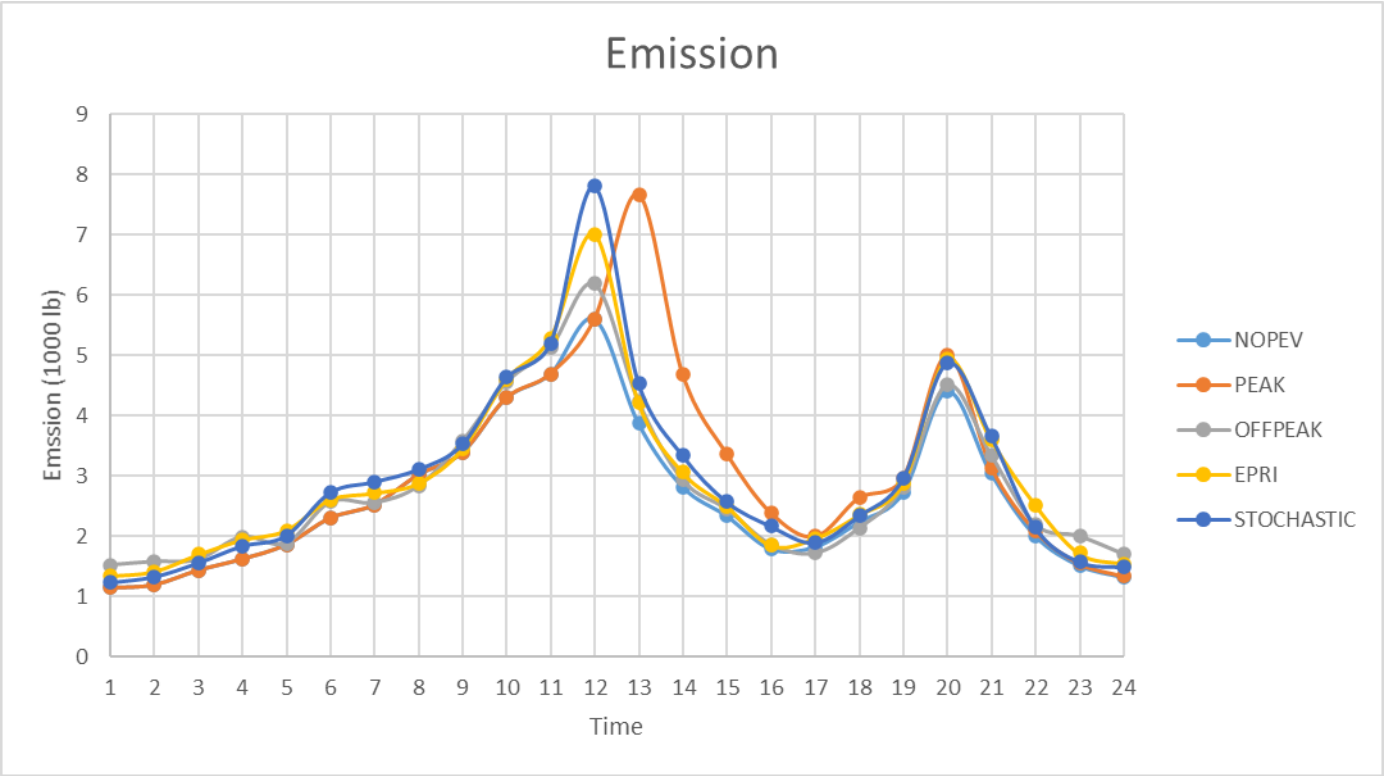


**Fig. 43** Plot of emission over 24 hours considering PEVs.

Considering the four charging scenarios above and data from figures (41, 42, 43), the off-peak charging scenario outperforms the other scenarios in terms of cost and emission being $716560 and 67922 lb, respectively which shows how it has the advantage in reducing cost and pollutant emissions.

## Conclusion

This report presents and compares two main techniques (PSO & GA) on MATLAB to find a global minimum for a given search space. According to the analysis performed, (PSO) outperformed the other method in terms of high-quality solution, less computations per iteration, consistency, and accuracy. Where (PSO) method was able to achieve a greater global minimum compared to genetic algorithm, while also utilizing iterations less than the amount used by genetic algorithm and each iteration was computed at a much faster speed compared to genetic algorithm. Thus, the proposed global minimum finding approach was used to find the best solution for cost and emission minimization over different charge scenarios to control the charging of a large community fleet of PEVs. Using the formulated (PSO) technique to perform the DEED on a 10-unit test system, the results provided a dispatch plan as shown on tables (6.9, 6.16, 6.23, 6.30, 6.37). The dispatch plans showed that the impact of PEVs on the load demand considering costs and emissions as shown in figures (41, 42, 43) was at minimum during off-peak charging scenario. Thus, charging time of electric vehicles would be best suited over hours from 23:00 to 6:00.

# References


[1] Yang, Z., Li, K., Niu, Q., Xue, Y., & Foley, A. (2014). A self-learning TLBO based dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads. Modern power systems, 2(4), 298-307. Retrieved 11 6, 2020, from https://link.springer.com/article/10.1007/s40565-014-0087-6

[2] Zhu, J. (n.d.). Optimization of Power System Operation. Wiley-IEEE Press. Retrieved 11 6, 2020

[3] S. Batmunkh and Z. Battogtokh, "The exhausting pollutants from coal combustion of Fourth Thermal Power Plant of Ulaanbaatar," 2007 International Forum on Strategic Technology, Ulaanbaatar, 2007, pp. 158-161, doi: 10.1109/IFOST.2007.4798548.

[4] F. N. Al Farsi, M. H. Albadi, N. Hosseinzadeh and A. H. Al Badi, "Economic Dispatch in power systems," 2015 IEEE 8th GCC Conference & Exhibition, Muscat, 2015, pp. 1-6, doi: 10.1109/IEEEGCC.2015.7060068.

[5] A. Lambora, K. Gupta and K. Chopra, "Genetic Algorithm- A Literature Review," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 380-384, doi: 10.1109/COMITCon.2019.8862255.

[6] K. B. Sahay, A. Sonkar and A. Kumar, "Economic Load Dispatch Using Genetic Algorithm Optimization Technique," 2018 International Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE), Phuket, Thailand, 2018, pp. 1-5, doi: 10.23919/ICUE-GESD.2018.8635729.

[7] Y. Xiaoyan, G. Chunlin, X. Xuan, H. Dequan and M. Zhou, "Research on large scale electric vehicles participating in the economic dispatch of wind and thermal power system," 2017 China International Electrical and Energy Conference (CIEEC), Beijing, 2017, pp. 223-228, doi: 10.1109/CIEEC.2017.8388450.

[8] Han, X. & Gooi, H.B. & Kirschen, D.s. (2001). Dynamic Economic Dispatch: Feasible and Optimal Solutions. Power Engineering Review, IEEE. 21. 56-56. 10.1109/MPER.2001.4311288.

[9] Mohammadi-Ivatloo B, Rabiee A, Soroudi A et al (2012) Iteration PSO with time varying acceleration coefficients for solving non-convex economic dispatch problems. Int J Electr Power Energy Syst 42(1):508–516

[10] Yuan XH, Wang L, Zhang YC et al (2009) A hybrid differential evolution method for dynamic economic dispatch with valvepoint effects. Expert Syst Appl 36(2–2):4042–4048

[11] Niu Q, Zhang HY, Wang XH et al (2014) A hybrid harmony search with arithmetic crossover operation for economic dispatch. Int J Electr Power Energy Syst 62:237–257

[12] Wang LF, Singh C (2007) Environmental/economic power dispatch using a fuzzified multi-objective particle swarm optimization algorithm. Electr Power Syst Res 77(12):1654–1664

[13] EPRI executive summary: environmental assessment of plug-in hybrid electric vehicles, Volume 2: United States air quality analysis based on AEO-2006 assumptions for 2030 (2007). Electric Power Research Institute, Palo Alto

[14] Foley A, Tyther B, Calnan P et al (2013) Impacts of electric vehicle charging under electricity market operations. Appl Energy 101:93–102

[15] Niu Q, Zhang HY, Li K et al (2014) An efficient harmony search with new pitch adjustment for dynamic economic dispatch. Energy 65:25–43

[16] Zhang Y, Yao F, Lu HHC et al (2013) Sequential quadratic programming particle swarm optimization for wind power system operations considering emissions. J Mod Power Syst Clean Energy 1(3):231–240

[17] Souag, Slimane & Benhamida, Farid & Belhachem, R.. (2013). Dynamic Economic Dispatch using an Improved Quadratic Programming Method.

[18] Beasley, John. (1999). Heuristic Algorithms for the Unconstrained Binary Quadratic Programming Problem. London, England.

[19] Xia, Xiaohua & Elaiw, A.. (2010). Optimal dynamic economic dispatch of generation: A review. Electric Power Systems Research. 80. 975-986. 10.1016/j.epsr.2009.12.012.

[20] X. Tang (2013). Economic scheduling for power systems with significant wind generation. Belfast, Northern Ireland.

[21] M. Basu (2006) Particle Swarm Optimization Based Goal-Attainment Method for Dynamic Economic Emission Dispatch, Electric Power Components and Systems, 34:9, 1015-1025, DOI: 10.1080/15325000600596759

[22] Razali, Noraini & Geraghty, John. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. 2.

[23] P. Sara, H. David, B. Paul. (2019, December). “Net zero in the UK”. House of Commons Library [Online]. Briefing Paper CBP8590. Available: https://commonslibrary.parliament.uk/research-briefings/cbp-8590/
[24] Z. Runsen, F. Shinichiro. (2020). “The role of transport electrification in global climate change mitigation scenarios”. Environ. Res. Lett. 15 034019 [online]. Available: https://iopscience.iop.org/article/10.1088/1748-9326/ab6658
[25] "Adoption of the Paris Agreement," United Nations, New York, 2015.
[26] "Climate Change and CO2," Int. Organization of Motor Vehicle Manufacturers, Paris, 2008.
[27] "Technology Road Map: Electric and Plug in Hybrid Electric Vehicles," International Energy Agency Report, 2009; updated 2011.
[28] FOLEY, A. , & Ó GALLACHÓIR, B. (2015). Analysis of electric vehicle charging using the traditional generation expansion planning analysis tool WASP-IV. J. Mod. Power Syst. Clean Energy, 3 (2). doi: 10.1007/s40565-015-0126-y
[29] S. Negarestani, A. R. Ghahnavieh and A. S. Mobarakeh, "A study of the reliability of various types of the electric vehicles," 2012 IEEE International Electric Vehicle Conference, 2012, pp. 1-6, doi: 10.1109/IEVC.2012.6183250.
[30] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[31] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350
[32] M. Nour, H. Ramadan, A. Ali and C. Farkas, "Impacts of plug-in electric vehicles charging on low voltage distribution network," 2018 International Conference on Innovative Trends in Computer Engineering (ITCE), 2018, pp. 357-362, doi: 10.1109/ITCE.2018.8316650.
[33] C. Madrid, J. Argueta, and J. Smith, “Performance characterization—1999 Nissan Altra-EV with lithium-ion battery,” 1999. Retrieved 3 11, 2020 from http://74.121.199.247/sites/default/files/pdf/fsev/altra_report.pdf
[34] N. B. Mohd Shariff, M. Al Essa and L. Cipcigan, "Probabilistic analysis of electric vehicles charging load impact on residential Distributions Networks," 2016 IEEE International Energy Conference (ENERGYCON), Leuven, Belgium, 2016, pp. 1-6, doi: 10.1109/ENERGYCON.2016.7513943.
[35] S. A. Kazarlis, A. G. Bakirtzis and V. Petridis, "A genetic algorithm solution to the unit commitment problem," in *IEEE Transactions on Power Systems*, vol. 11, no. 1, pp. 83-92, Feb. 1996, doi: 10.1109/59.485989.
[36] A. Kumar, M. Bhadu and H. Kumawat, "Unit commitment in thermal power generation dispatching with integration of PHEVs," 2018 8th IEEE India International Conference on Power Electronics (IICPE), Jaipur, India, 2018, pp. 1-6, doi: 10.1109/IICPE.2018.8709614.

# Appendices

Appendix A. Quadprog code

```
% ELEC3875 Individual Project
% School of Electronic & Electrical Engineering
% University of Leeds
% 2020/21
%
% Name: Abdullah Essa
% Username: el18ae
% Student ID Number: 201256467
% Date: 10/12/2020
%
% Using Quadprog Function for 5 Generation Units and 500MW Demand Supervised by
Professor Kang Li & Mingjia Yin
clear; clc;
% SED Cost Function given as:
% F = 0.008P1^2 +2P1 + 0.003P2^2 + 1.8P2 +0.0012P3^2 + 2.1P3 + 0.001P4^2 + 2P4 +
0.0015P5^2 +1.8P5 + 245

% where size of matrix H is (ixi) and cost coefficient (c) of Pi is at H(i,i)
H = 2*[0.008 0 0 0 0; 0 0.003 0 0 0; 0 0 0.0012 0 0; 0 0 0 0.001 0; 0 0 0 0 0.0015];

% where size of matrix f is (1xi) and cost coefficient (b) of Pi is at f(1,i)
f = [2; 1.8; 2.1; 2; 1.8];

% equality constraint being P1 + P2 + P3 + P4 + P5 = 500 MW
A_eq = [1 1 1 1 1];             % coefficient of each Pi at A_eq(i,1)
B_eq = [500];                   % result of each equality constraint
ub = [75; 125; 175; 250; 300];  % upper bound of each Pi unit
lb = [10; 20; 30; 40; 50];      % lower bound of each Pi unit

[X,J] = quadprog(H,f,[],[],A_eq,B_eq,lb,ub);

% quadprog function does not consider constant (a) of each Pi generation
% unit, thus needs to be added to final total cost.

min_total_cost = J + 245
```

Appendix B. Genetic Algorithm code

```
% ELEC3875 Individual Project
```

```
% School of Electronic & Electrical Engineering
% University of Leeds
% 2020/21
%
% Name: Abdullah Essa
% Username: el18ae
% Student ID Number: 201256467
% Date: 19/11/2020
%
% Genetic Algorithm Code Supervised by Professor Kang Li & Mingjia Yin

% code starts here:
clear; clc;
%rng(100, 'twister');

%======================INITIALIZATION=======================%
generations = 100;                                  % number of generations
Np = 300;                                           % population number (# of solutions)
bits = 16;                                            % number of resolution bits
gen_units = 4;                                       % number of generation units
population = randi([0 1], Np, (bits*gen_units));     % random binary generation of
population solutions
lowerLimit = [10 20 30 40];                              % place lower limits of
gen_units(1),...,gen_units(N-1)
upperLimit = [75 125 175 250];                              % place lower limits of
gen_units(1),...,gen_units(N-1)
lowerLimit_N = 50;                                  % lower limit for final slack
generator
upperLimit_N = 300;                                  % upper limit for final slack
generator
h = [1 100];                                        % penalty factors for final generator
DV = zeros(Np, gen_units);                            % decoded value
RV = zeros(Np, gen_units);                            % real value
PF = zeros(Np,1);                                     % fitness value vector
Pc = 0.9;                                           % cross-over probability
offspring = zeros(Np, (bits * gen_units));           % setting up the matrix that will
hold the new offspring solutions
Pm = 0.1;                                             % mutation probability
OF = zeros(Np,1);                                     % offspring fitness vector
a = [0.0080 0.0030 0.0012 0.0010 0.0015];                          % coefficient (an) of
in-out characteristic equation
b = [2 1.8 2.1 2 1.8];                                   % coefficient (bn) of in-out
characteristic equation
c = [25 60 100 120 40];                                          % coefficient (cn) of
in-out characteristic equation
P_demand = 500;                                       % power demand
%============================================================%
COUNTER = 0;
%======================GENERATIONS LOOP======================%
for T = 1:generations

%======================GETTING FITNESS=======================%
% I want to be able to decode any amount of generation units
for i1 = 1:gen_units
    for j1 = 1:Np

        % dynamic row and column allocations to determine DV and RV
        DV(j1,i1) = bi2de(population(j1, ((1 + bits * (i1 - 1)):(bits * i1))));
        RV(j1,i1) = lowerLimit(i1) + (((upperLimit(i1) - lowerLimit(i1)) / ((2^bits) -
1)) * DV(j1,i1));

    end

end
```

```
% getting the fitness
    for k1 = 1:Np

        F1 = (a(1)*(RV(k1,1))^2 + b(1)*(RV(k1,1)) + c(1));
% generator 1 in-out characteristics
        F2 = (a(2)*(RV(k1,2))^2 + b(2)*(RV(k1,2)) + c(2));
% generator 2 in-out characteristics
        F3 = (a(3)*(RV(k1,3))^2 + b(3)*(RV(k1,3)) + c(3));
% generator 3 in-out characteristics
        F4 = (a(4)*(RV(k1,4))^2 + b(4)*(RV(k1,4)) + c(4));
% generator 4 in-out characteristics
        F5 = (a(5)*(P_demand-RV(k1,1)-RV(k1,2)-RV(k1,3)-RV(k1,4))^2 + b(5)*(P_demand-
RV(k1,1)-RV(k1,2)-RV(k1,3)-RV(k1,4)) + c(5));  % generator 5 in-out characteristics
        penalty1 = h(1)*(upperLimit_N-(P_demand-RV(k1,1)-RV(k1,2)-RV(k1,3)-
RV(k1,4)))^2;                                          % penalty for generator N max
constraint
        penalty2 = h(2)*((P_demand-RV(k1,1)-RV(k1,2)-RV(k1,3)-RV(k1,4))-
lowerLimit_N)^2;                                        % penalty for generator N
min constraint
        PF(k1) = (F1 + F2 + F3 + F4 + F5 + penalty1 + penalty2);
% fitness function

    end

%================================================================%

%====================PARENT CHOOSE============================%
% tournament method for the fitness values to choose parents for cross-over
% procedure.
participant_selects = zeros(Np,1);      % matrix for selected winners
participant_index = randperm(Np);       % shuffling the participants positions

% ensure each participant to fight atleast twice
for z = 1:Np-1

    % assign and store their fitness
    CF = [PF(participant_index(z),1) PF(participant_index(z+1),1)];

    % determine winner
    winner = min(CF);

    % retrieve and store the index of the winner
    for search1 = 1:Np

        if winner == PF(search1)

            participant_selects(z,1) = search1;

        end

    end

end

% the last two participants fight
CF = [PF(participant_index(Np),1) PF(participant_index(1),1)];     % assign and store
their fitness
winner = min(CF);                                                  % determine winner

% retrieve and store the index of the winner
    for search2 = 1:Np
```

```
        if winner == PF(search2)

            participant_selects(Np,1) = search2;

        end

    end
%=============================================================%

%==================SINGLE-POINT_CROSS-OVER====================%
% we need to get the population solutions using the winner indexes

for q = 1:2:Np-1

    if randsrc(1,1,[1,2;Pc,(1-Pc)]) == 1                                % Pc*100%
chance for (1); cross-over

        % random cross-over point generation
        crossPoint = randi([1 ((bits*gen_units))], 1, 1);             % we swap
positions from chosen element location

        % offspring (1)
        offspring(q,1:(crossPoint-1)) =
population(participant_selects(q,1),1:(crossPoint-1));
        offspring(q,crossPoint:(bits*gen_units)) =
population(participant_selects(q+1,1),crossPoint:(bits*gen_units));

        % offspring (2)
        offspring(q+1,1:(crossPoint-1)) =
population(participant_selects(q+1,1),1:(crossPoint-1));
        offspring(q+1,crossPoint:(bits*gen_units)) =
population(participant_selects(q,1),crossPoint:(bits*gen_units));


    elseif randsrc(1,1,[1,2;Pc,(1-Pc)]) == 2                            % (1-Pc)*100%
chance for (2); no cross-over

        offspring(q,:) = population(participant_selects(q,1),:);        % first
offspring
        offspring(q+1,:) = population(participant_selects(q+1,1),:);    % second
offspring

    end

end
%=============================================================%

%========================MUTATION=============================%
for m1 = 1:Np  % loops rows

    for m2 = 1:(bits*gen_units)  % loops columns

        if randsrc(1,1,[1,2;Pm,(1-Pm)]) == 1

            offspring(m1,m2) = not(offspring(m1,m2));            % bit flip

        else

            offspring(m1,m2) = offspring(m1,m2);                 % keep the same
```

```
        end

    end

end
%=============================================================%

%=====================OFFSPRING FITNESS=====================%
% I want to be able to decode any amount of generation units
for i2 = 1:gen_units
    for j2 = 1:Np

        % dynamic row and column allocations to determine DV and RV
        DV(j2,i2) = bi2de(offspring(j2, ((1 + bits * (i2 - 1)):(bits * i2))));
        RV(j2,i2) = lowerLimit(i2) + (((upperLimit(i2) - lowerLimit(i2)) / ((2^bits) -
1)) * DV(j2,i2));

    end

end

% getting the fitness
    for k2 = 1:Np

        FF1 = (a(1)*(RV(k2,1))^2 + b(1)*(RV(k2,1)) + c(1));
% generator 1 in-out characteristics
        FF2 = (a(2)*(RV(k2,2))^2 + b(2)*(RV(k2,2)) + c(2));
% generator 2 in-out characteristics
        FF3 = (a(3)*(RV(k2,3))^2 + b(3)*(RV(k2,3)) + c(3));
% generator 3 in-out characteristics
        FF4 = (a(4)*(RV(k2,4))^2 + b(4)*(RV(k2,4)) + c(4));
% generator 4 in-out characteristics
        FF5 = (a(5)*(P_demand-RV(k2,1)-RV(k2,2)-RV(k2,3)-RV(k2,4))^2 + b(5)*(P_demand-
RV(k2,1)-RV(k2,2)-RV(k2,3)-RV(k2,4)) + c(5));  % generator 5 in-out characteristics
        Ppenalty1 = h(1)*(upperLimit_N-(P_demand-RV(k2,1)-RV(k2,2)-RV(k2,3)-
RV(k2,4)))^2;                                                % penalty for generator N max
constraint
        Ppenalty2 = h(2)*((P_demand-RV(k2,1)-RV(k2,2)-RV(k2,3)-RV(k2,4))-
lowerLimit_N)^2;                                               % penalty for generator N
min constraint
        OF(k2) = (FF1 + FF2 + FF3 + FF4 + FF5 + Ppenalty1 + Ppenalty2);
% fitness function

    end
%=============================================================%

%=====================COMBINING FITNESS=====================%
combinedFitness = zeros(2*Np,1);
combinedSolution = zeros(2*Np,(bits * gen_units));

for p1 = 1:2*Np

    if p1 < Np+1

        % inserting parent fitness
        combinedFitness(p1,1) = PF(participant_selects(p1),1);
        % inserting parent solutions
        combinedSolution(p1,1:(bits * gen_units)) =
population(participant_selects(p1),1:(bits * gen_units));

    elseif p1 > Np
```

```
        % inserting offspring fitness
        combinedFitness(p1,1) = OF(p1-Np,1);
        % inserting offspring solutions
        combinedSolution(p1,1:(bits * gen_units)) = offspring(p1-Np,1:(bits *
gen_units));
    end

end

%=============================================================%

%====================SORTING FITNESS==========================%

sortCombinedFitness = sort(combinedFitness);                % sort the fitness
(smallest first)
chosenFitness = zeros(Np,1);                                % store chosen fitness
chosenCombinedSolution = zeros(Np,(bits * gen_units));      % store chosen solution

for p2 = 1:Np       % take only the first Np best solutions

    chosenFitness(p2,1) = sortCombinedFitness(p2,1);

    for search3 = 1:Np*2

        if search3 < Np+1

            if chosenFitness(p2,1) == PF(search3,1)     % check through population
fitness

                chosenCombinedSolution(p2,1:(bits*gen_units)) =
population(search3,1:(bits*gen_units));

            end

        else

            if chosenFitness(p2,1) == OF(search3-Np,1)

                chosenCombinedSolution(p2,1:(bits*gen_units)) = offspring(search3-
Np,1:(bits*gen_units));

            end

        end

    end

end
%=============================================================%

%================ASSIGNING NEW GENERATION====================%

for o = 1:Np

    population(o,1:(bits*gen_units)) = 0;
    population(o,1:(bits*gen_units)) = chosenCombinedSolution(o,1:(bits*gen_units));

end

%=============================================================%
```

```
%====================RESET COUNTERS============================%

i1 = 1;
j1 = 1;
k1 = 1;
z = 1;
search1 = 1;
search2 = 1;
search3 = 1;
q = 1;
m1 = 1;
m2 = 1;
i2 = 1;
j2 = 1;
k2 = 1;
p1 = 1;
p2 = 1;
o = 1;

%==============================================================%

%===================CONVERGENCE PLOT===========================%

    % plotting
    x_axis(T,1) = T;                    % counts and stores the number of generations
    y_axis(T,1) = chosenFitness(1,1);            % minimum fitness of each generation
    % plot(x_axis,y_axis,'.')
    % hold on

%==============================================================%

%==============================================================%
    clc;
    COUNTER = COUNTER +1;                                % counts loop number
    GA_PERCENT_DONE = (COUNTER / generations)*100    % percentage done from code
    population;

end
% end of generations for loop
%==============================================================%

% real final values gen1 until genN-1
for i1 = 1:gen_units
    for j1 = 1:Np

        % dynamic row and column allocations to determine DV and RV
        DV(j1,i1) = bi2de(population(j1, ((1 + bits * (i1 - 1)):(bits * i1))));
        RV(j1,i1) = lowerLimit(i1) + (((upperLimit(i1) - lowerLimit(i1)) / ((2^bits) -
1)) * DV(j1,i1));

    end

end

plot(x_axis,y_axis,'.')
title('Plot of Fitness Per Iteration')
xlabel('Iterations')
ylabel('Fitness')

RV;  % showing results of last generation of real values to check
```

```
result(1,1:5) = [RV(1,1), RV(1,2), RV(1,3), RV(1,4), (P_demand - RV(1,1) - RV(1,2) -
RV(1,3) - RV(1,4))]
```

## Appendix C. Particle Swarm Optimization code

```
%% initializing parameters
population = 20000;          % number of iterations and/or particles
iter = 300;                 % number of iterations
D = 10;                      % number of generation units
lb = [150 150 20 20 25 20 25 10 10];          % lower bound of the generation units
range of operation
ub = [450 455 130 130 162 80 85 55 55];       % upper bound of the generation units
range of operation
lb_lastgen = 50;             % lower bound of the reference generation unit range of
operation
```

```
ub_lastgen = 300;            % upper bound of the reference generation unit range of
operation
lb1 = [150 150 20 20 25 20 25 10 10 50];      % lower bound of the generation units
range of operation (all)
ub1 = [450 455 130 130 162 80 85 55 55 300];  % upper bound of the generation units
range of operation (all)
UR = [91 91 40 50 50 20 20 11 11 50];       % ramp up limits
DR = [91 91 40 50 50 20 20 11 11 50];       % ramp down limits

PD_day = [861.34 911.34 928.489 1028.489 1034.884 1134.884 1150 1200 1300 1400 1450
1500 1400 1300 1200 1050 1000 1100 1200 1400 1300 1100 1061.34 961.34];  % power demand
MW

% coefficients used for cost equation
a = [0.00173 0.00031 0.002 0.00211 0.00211 0.00712 0.00079 0.00413 0.00222 0.00048];
b = [27.79 17.26 16.6 16.5 19.7 22.26 27.74 25.92 27.27 16.19];
c = [670 970 700 680 450 370 480 660 685 1000];

% coefficients used for emission equation
gamma = [0.0180 0.0150 0.0105 0.0080 0.0120 0.0180 0.0150 0.0105 0.0080 0.0120];
beta = [-0.805 -0.555 -1.355 -0.6 -0.555 -0.805 -0.555 -1.355 -0.6 -0.555];
alpha = [80 50 60 45 30 80 50 60 45 30];
eta=[0.655 0.5773 0.4968 0.486 0.5035 0.655 0.5773 0.4968 0.486 0.5035];
delta=[0.02846 0.02446 0.02270 0.01948 0.02075 0.02846 0.02446 0.02270 0.01948
0.02075];

c1=0;  %acceleration coefficients
c2=1;  %acceleration coefficients
w=1;  %intertia weight maximum
wee = 0.5;    % weighting factor

% fitness vector
fit_vector = zeros(24,1);
% solution matrix
sol_matrix = zeros(24,D);
% cost matrix (for each generator)
cost = zeros(24,D);
% cost matrix (for each generator)
emission = zeros(24,D);
% hour counter initializer
mm = 1;
% matrix to store positions of each generation unit with regards to their final fitness
final_position = zeros(24,1);
% matrix to store cost of each generation unit with regards to their final fitness
final_cost = zeros(24,1);
% matrix to store emission of each generation unit with regards to their final fitness
final_emission = zeros(24,1);


%% for loop to iterate through every hour (24 hours), taking the value of PD
for k = 1:24
w=0.8;  %intertia weight maximum reset
clc;
percent_done = k/24 * 100
PD = PD_day(k);  % taking the power demand for the specific hour
%% initializing matrix of position
    if mm == 1
        X = repmat(lb,population,1) + repmat((ub-lb),population,1).*rand(population,D-
1);  % create random positions (power)for non-reference units

        % check each particle if it meets the condition for the while loop
        for p = 1:population
```

```
            while ((PD - sum(X(p,1:9))) > ub_lastgen) || ((PD - sum(X(p,1:9))) <
lb_lastgen)
                % create random position again
                X(p,:) = repmat(lb,1,1) + repmat((ub-lb),1,1).*rand(1,D-1);
            end
        end

    elseif mm > 1
        % make sure random positions within ramp up and down limits
        LB_constrained = sol_matrix(mm-1,:) - DR;
        UB_constrained = sol_matrix(mm-1,:) + UR;

        % make sure new constraint due to ramp limits are within generator constraints
        LB_constrained = max(LB_constrained,lb1);
        UB_constrained = min(UB_constrained,ub1);
        X = repmat(LB_constrained(:,1:9),population,1) + repmat((UB_constrained(:,1:9)-
LB_constrained(:,1:9)),population,1).*rand(population,D-1);
        %X = repmat(lb,population,1) + repmat((ub-lb),population,1).*rand(population,D-
1);  % create random positions (power)for non-reference units

        % check each particle if it meets the condition for the while loop
        for pp = 1:population
            while ((PD - sum(X(pp,1:9))) > UB_constrained(1,D)) || ((PD -
sum(X(pp,1:9))) < LB_constrained(1,D))
                % create random position again
                X(pp,:) = repmat(LB_constrained(:,1:9),1,1) +
repmat((UB_constrained(:,1:9)-LB_constrained(:,1:9)),1,1).*rand(1,D-1);
            end
        end
    end

    % getting the reference unit
    for i = 1:population
        X(i,D) = PD - (X(i,1) + X(i,2) + X(i,3) + X(i,4) + X(i,5) + X(i,6) + X(i,7) +
X(i,8) + X(i,9));  % reference unit position
        if mm == 1  % for the first hour no ramp constraints
            X(i,D) = max(X(i,D),lb_lastgen);
            X(i,D) = min(X(i,D),ub_lastgen);
        elseif mm > 1  % vary the limitation to match with the ramp constraints
            X(i,D) = max(X(i,D),LB_constrained(1,D));
            X(i,D) = min(X(i,D),UB_constrained(1,D));
        end
    end

%% initializing matrix of velocities
if mm == 1  % for the first hour no ramp constraints
    V = repmat(lb1,population,1) + repmat((ub1-lb1),population,1).*rand(population,D);
% create random velocities
elseif mm > 1
    V = repmat(LB_constrained(1,1:10),population,1) + repmat((UB_constrained(1,1:10)-
LB_constrained(1,1:10)),population,1).*rand(population,D);
end

%% evaluating initial fitness
F_cost = zeros(population,1);      % the cost fitness storage
F_emission = zeros(population,1);  % emission fitness storage
F = zeros(population,1);           % combined fitness storage
for i = 1:population
    % economic fitness
    F1_cost = (a(1)*(X(i,1))^2 + b(1)*(X(i,1)) + c(1));  % gen1 fitness
    F2_cost = (a(2)*(X(i,2))^2 + b(2)*(X(i,2)) + c(2));  % gen2 fitness
    F3_cost = (a(3)*(X(i,3))^2 + b(3)*(X(i,3)) + c(3));  % gen3 fitness
    F4_cost = (a(4)*(X(i,4))^2 + b(4)*(X(i,4)) + c(4));  % gen4 fitness
    F5_cost = (a(5)*(X(i,5))^2 + b(5)*(X(i,5)) + c(5));  % gen5 fitness
```

```
    F6_cost = (a(6)*(X(i,6))^2 + b(6)*(X(i,6)) + c(6));  % gen6 fitness
    F7_cost = (a(7)*(X(i,7))^2 + b(7)*(X(i,7)) + c(7));  % gen7 fitness
    F8_cost = (a(8)*(X(i,8))^2 + b(8)*(X(i,8)) + c(8));  % gen8 fitness
    F9_cost = (a(9)*(X(i,9))^2 + b(9)*(X(i,9)) + c(9));  % gen9 fitness
    F10_cost = (a(D)*(X(i,D))^2 + b(D)*(X(i,D)) + c(D)); % reference unit fitness
    F_cost(i,1) = F1_cost + F2_cost + F3_cost + F4_cost + F5_cost + F6_cost + F7_cost +
F8_cost + F9_cost + F10_cost;  % cost fitness

    % emission fitness
    F1_emission = (gamma(1)*(X(i,1))^2 + beta(1)*(X(i,1)) + alpha(1)) +
eta(1)*exp(delta(1)*X(i,1));  % gen1 fitness
    F2_emission = (gamma(2)*(X(i,2))^2 + beta(2)*(X(i,2)) + alpha(2)) +
eta(2)*exp(delta(2)*X(i,2));  % gen2 fitness
    F3_emission = (gamma(3)*(X(i,3))^2 + beta(3)*(X(i,3)) + alpha(3)) +
eta(3)*exp(delta(3)*X(i,3));  % gen3 fitness
    F4_emission = (gamma(4)*(X(i,4))^2 + beta(4)*(X(i,4)) + alpha(4)) +
eta(4)*exp(delta(4)*X(i,4));  % gen4 fitness
    F5_emission = (gamma(5)*(X(i,5))^2 + beta(5)*(X(i,5)) + alpha(5)) +
eta(5)*exp(delta(5)*X(i,5));  % gen5 fitness
    F6_emission = (gamma(6)*(X(i,6))^2 + beta(6)*(X(i,6)) + alpha(6)) +
eta(6)*exp(delta(6)*X(i,6));  % gen6 fitness
    F7_emission = (gamma(7)*(X(i,7))^2 + beta(7)*(X(i,7)) + alpha(7)) +
eta(7)*exp(delta(7)*X(i,7));  % gen7 fitness
    F8_emission = (gamma(8)*(X(i,8))^2 + beta(8)*(X(i,8)) + alpha(8)) +
eta(8)*exp(delta(8)*X(i,8));  % gen8 fitness
    F9_emission = (gamma(9)*(X(i,9))^2 + beta(9)*(X(i,9)) + alpha(9)) +
eta(9)*exp(delta(9)*X(i,9));  % gen9 fitness
    F10_emission = (gamma(D)*(X(i,D))^2 + beta(D)*(X(i,D)) + alpha(D)) +
eta(D)*exp(delta(D)*X(i,D)); % reference unit fitness
    F_emission(i,1) = F1_emission + F2_emission + F3_emission + F4_emission +
F5_emission + F6_emission + F7_emission + F8_emission + F9_emission + F10_emission;  %
emission fitness

    % combined fitness
     F(i,1) = (wee)*F_cost(i,1) + (1 - wee)*F_emission(i,1);  % combined fitness

end

pbest = X;                          % initializing the personal best solutions
fit_pbest = F;                      % storing the fitness of the personal best
solutions
[F_gbest, index] = min(fit_pbest);  % determining the global best fitness
gbest = X(index,:);                 % storing the global best fitness power positions

%% PSO
for i = 1:iter
    w = w - (w/iter) * i;  % new weight
   for j = 1:population
      % velocity/position determination
      V(j,:) = w*V(j,:) + c1*rand(1,D).*(pbest(j,:)-X(j,:))+c2*rand(1,D).*(gbest-
X(j,:));  % new velocity
      X(j,:) = X(j,:) + V(j,:);  % new position

      if mm == 1  % for the first hour no ramp constraints
        X(j,:) = max(X(j,:),lb1);
        X(j,:) = min(X(j,:),ub1);
      elseif mm > 1  % vary the limitation to match with the ramp constraints
        X(j,:) = max(X(j,:),LB_constrained);
        X(j,:) = min(X(j,:),UB_constrained);
      end


      % determine the fitness of new positions
```

```
        % economic fitness
        F1_cost = (a(1)*(X(j,1))^2 + b(1)*(X(j,1)) + c(1));  % gen1 fitness
        F2_cost = (a(2)*(X(j,2))^2 + b(2)*(X(j,2)) + c(2));  % gen2 fitness
        F3_cost = (a(3)*(X(j,3))^2 + b(3)*(X(j,3)) + c(3));  % gen3 fitness
        F4_cost = (a(4)*(X(j,4))^2 + b(4)*(X(j,4)) + c(4));  % gen4 fitness
        F5_cost = (a(5)*(X(j,5))^2 + b(5)*(X(j,5)) + c(5));  % gen5 fitness
        F6_cost = (a(6)*(X(j,6))^2 + b(6)*(X(j,6)) + c(6));  % gen6 fitness
        F7_cost = (a(7)*(X(j,7))^2 + b(7)*(X(j,7)) + c(7));  % gen7 fitness
        F8_cost = (a(8)*(X(j,8))^2 + b(8)*(X(j,8)) + c(8));  % gen8 fitness
        F9_cost = (a(9)*(X(j,9))^2 + b(9)*(X(j,9)) + c(9));  % gen9 fitness
        F10_cost = (a(D)*(X(i,D))^2 + b(D)*(X(i,D)) + c(D)); % reference unit fitness
        F_cost(j,1) = F1_cost + F2_cost + F3_cost + F4_cost + F5_cost + F6_cost +
F7_cost + F8_cost + F9_cost + F10_cost;  % cost fitness

        % emission fitness
        F1_emission = (gamma(1)*(X(j,1))^2 + beta(1)*(X(j,1)) + alpha(1)) +
eta(1)*exp(delta(1)*X(j,1));  % gen1 fitness
        F2_emission = (gamma(2)*(X(j,2))^2 + beta(2)*(X(j,2)) + alpha(2)) +
eta(2)*exp(delta(2)*X(j,2));  % gen2 fitness
        F3_emission = (gamma(3)*(X(j,3))^2 + beta(3)*(X(j,3)) + alpha(3)) +
eta(3)*exp(delta(3)*X(j,3));  % gen3 fitness
        F4_emission = (gamma(4)*(X(j,4))^2 + beta(4)*(X(j,4)) + alpha(4)) +
eta(4)*exp(delta(4)*X(j,4));  % gen4 fitness
        F5_emission = (gamma(5)*(X(j,5))^2 + beta(5)*(X(j,5)) + alpha(5)) +
eta(5)*exp(delta(5)*X(j,5));  % gen5 fitness
        F6_emission = (gamma(6)*(X(j,6))^2 + beta(6)*(X(j,6)) + alpha(6)) +
eta(6)*exp(delta(6)*X(j,6));  % gen6 fitness
        F7_emission = (gamma(7)*(X(j,7))^2 + beta(7)*(X(j,7)) + alpha(7)) +
eta(7)*exp(delta(7)*X(j,7));  % gen7 fitness
        F8_emission = (gamma(8)*(X(j,8))^2 + beta(8)*(X(j,8)) + alpha(8)) +
eta(8)*exp(delta(8)*X(j,8));  % gen8 fitness
        F9_emission = (gamma(9)*(X(j,9))^2 + beta(9)*(X(j,9)) + alpha(9)) +
eta(9)*exp(delta(9)*X(j,9));  % gen9 fitness
        F10_emission = (gamma(D)*(X(j,D))^2 + beta(D)*(X(j,D)) + alpha(D)) +
eta(D)*exp(delta(D)*X(j,D)); % reference unit fitness
        F_emission(j,1) = F1_emission + F2_emission + F3_emission + F4_emission +
F5_emission + F6_emission + F7_emission + F8_emission + F9_emission + F10_emission;  %
emission fitness

        % combined fitness
        F(j,1) = (wee)*F_cost(j,1) + (1 - wee)*F_emission(j,1);  % combined fitness

      % check if there is a better personal best values
      if (F(j) < fit_pbest(j)) && (sum(X(j,:)) > 499.99) && (sum(X(j,:)) <= 500)
          fit_pbest(j) = F(j);
          pbest(j,:) = X(j,:);
      end

      % check if there is a better global best solution
      if (fit_pbest(j) < F_gbest) && (sum(gbest) > 499.99) && (sum(gbest) <= 500)
          F_gbest = fit_pbest(j);
          gbest = pbest(j,:);
      end
   end


end

%% finale
fit_vector(mm) = F_gbest;  % storing the last fitness for the hour
sol_matrix(mm,:) = gbest;  % storing the last solution for the hour
mm = mm + 1;
end
```

```
%% get total demand for 24 hours
 total_cost = 0;
 total_emission = 0;

 for i = 1:24
    % cost fitness
    cost(i,1) = (a(1)*(sol_matrix(i,1))^2 + b(1)*(sol_matrix(i,1)) + c(1));  % gen1
cost
    cost(i,2) = (a(2)*(sol_matrix(i,2))^2 + b(2)*(sol_matrix(i,2)) + c(2));  % gen2
cost
    cost(i,3) = (a(3)*(sol_matrix(i,3))^2 + b(3)*(sol_matrix(i,3)) + c(3));  % gen3
cost
    cost(i,4) = (a(4)*(sol_matrix(i,4))^2 + b(4)*(sol_matrix(i,4)) + c(4));  % gen4
cost
    cost(i,5) = (a(5)*(sol_matrix(i,5))^2 + b(5)*(sol_matrix(i,5)) + c(5));  % gen5
cost
    cost(i,6) = (a(6)*(sol_matrix(i,6))^2 + b(6)*(sol_matrix(i,6)) + c(6));  % gen6
cost
    cost(i,7) = (a(7)*(sol_matrix(i,7))^2 + b(7)*(sol_matrix(i,7)) + c(7));  % gen7
cost
    cost(i,8) = (a(8)*(sol_matrix(i,8))^2 + b(8)*(sol_matrix(i,8)) + c(8));  % gen8
cost
    cost(i,9) = (a(9)*(sol_matrix(i,9))^2 + b(9)*(sol_matrix(i,9)) + c(9));  % gen9
cost
    cost(i,D) = (a(D)*(sol_matrix(i,D))^2 + b(D)*(sol_matrix(i,D)) + c(D));  %
reference unit cost

    % emission fitness
    emission(i,1) = (gamma(1)*(sol_matrix(i,1))^2 + beta(1)*(sol_matrix(i,1)) +
alpha(1)) + eta(1)*exp(delta(1)*sol_matrix(i,1));  % gen1 fitness
    emission(i,2) = (gamma(2)*(sol_matrix(i,2))^2 + beta(2)*(sol_matrix(i,2)) +
alpha(2)) + eta(2)*exp(delta(2)*sol_matrix(i,2));  % gen2 fitness
    emission(i,3) = (gamma(3)*(sol_matrix(i,3))^2 + beta(3)*(sol_matrix(i,3)) +
alpha(3)) + eta(3)*exp(delta(3)*sol_matrix(i,3));  % gen3 fitness
    emission(i,4) = (gamma(4)*(sol_matrix(i,4))^2 + beta(4)*(sol_matrix(i,4)) +
alpha(4)) + eta(4)*exp(delta(4)*sol_matrix(i,4));  % gen4 fitness
    emission(i,5) = (gamma(5)*(sol_matrix(i,5))^2 + beta(5)*(sol_matrix(i,5)) +
alpha(5)) + eta(5)*exp(delta(5)*sol_matrix(i,5));  % gen5 fitness
    emission(i,6) = (gamma(6)*(sol_matrix(i,6))^2 + beta(6)*(sol_matrix(i,6)) +
alpha(6)) + eta(6)*exp(delta(6)*sol_matrix(i,6));  % gen6 fitness
    emission(i,7) = (gamma(7)*(sol_matrix(i,7))^2 + beta(7)*(sol_matrix(i,7)) +
alpha(7)) + eta(7)*exp(delta(7)*sol_matrix(i,7));  % gen7 fitness
    emission(i,8) = (gamma(8)*(sol_matrix(i,8))^2 + beta(8)*(sol_matrix(i,8)) +
alpha(8)) + eta(8)*exp(delta(8)*sol_matrix(i,8));  % gen8 fitness
    emission(i,9) = (gamma(9)*(sol_matrix(i,9))^2 + beta(9)*(sol_matrix(i,9)) +
alpha(9)) + eta(9)*exp(delta(9)*sol_matrix(i,9));  % gen9 fitness
    emission(i,D) = (gamma(D)*(sol_matrix(i,D))^2 + beta(D)*(sol_matrix(i,D)) +
alpha(D)) + eta(D)*exp(delta(D)*sol_matrix(i,D));  % reference unit fitness


    total_cost = total_cost + sum(cost(i,:));
    total_emission = total_emission + sum(emission(i,:));
    final_position(i) = sum(sol_matrix(i,:));
    final_cost(i) = sum(cost(i,:));
    final_emission(i) = sum(emission(i,:));
 end
```