Artificial Intelligence based Smart Grid scheduling algorithm using Reinforcement Learning

Project-II (CS57003)report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Dual Degree (Masters of Technology)

in

Computer Science and Engineering

by

Debajyoti Dasgupta

(18CS30051)

Under the supervision of

Prof. Partha Pratim Chakrabarti
Prof. Arijit Mondal

Computer Science and Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2022-23

April 26, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 26, 2023 (Debajyoti Dasgupta)

Place: Kharagpur (18CS30051)

COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Artificial Intelligence based Smart Grid scheduling algorithm using Reinforcement Learning**" submitted by **Debajyoti Dasgupta** (Roll No. 18CS30051) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Dual Degree (Masters of Technology) in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2022-23.

Date: April 26, 2023
Place: Kharagpur

Prof. Partha Pratim Chakrabarti
Prof. Arijit Mondal
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Debajyoti Dasgupta** Roll No: **18CS30051**

Degree for which submitted: **Dual Degree (Masters of Technology)**

Department: **Computer Science and Engineering**

Thesis title: **Artificial Intelligence based Smart Grid scheduling algorithm using Reinforcement Learning**

Thesis supervisor: **Prof. Partha Pratim Chakrabarti**

Prof. Arijit Mondal

Month and year of thesis submission: **April 26, 2023**

The power consumption of households has constantly been growing over the years Kahan 2019. To cope with this growth, intelligent management of the consumption profile of the households is necessary, such that the households can save electricity bills, and the stress on the power grid during peak hours can be reduced. However, implementing such a method is challenging due to the existence of randomness in the electricity price as well as the requirement and consumption of the appliances. To address these challenges, we propose a reinforcement learning-based AI agent for the demand response based on cost minimization for the client-side appliances while taking care of user satisfaction and reducing the peak load on the supply power. We propose a working architecture and method employed for training the models. Finally, we show that using Reinforcement Learning Approach, we can approach the lower bound to the optimized solution. We will use the Peccan Street dataset for consumption and synthetic data to simulate the time-of-the-day pricing of the electricity. We thus show using the dataset, that the Reinforcement learning model not only learns to minimize the cost of consumption while handling several devices but also provides a scalable architecture that will reduce the stress on the generation side. In this report, we also suggest two new modes of training the Reinforcement Learning Agent: incremental training and using a predictor network to improve training. We also give a comparative study of performance improvement in training using the newly proposed metrics. The study is then extended to a multi-agent environment where we build strong baselines to take account of human nature and show that careful modeling of the observation space can lead to efficient learning of agents in the cooperative scenario. Finally, the exploration continues with the results of the genetic algorithm-based solution.

Acknowledgements

I want to thank my mentors, Prof. Partha Pratim Chakrabarti and Prof. Arijit Mondal, for their exceptional guidance and support, without which this project would not have reached its full potential. They have always motivated me to explore as much as possible, look into numerous papers, and try as many ideas as possible. They have always supported me through whatever problems I faced during the project and resources. With their continuous input, the project has evolved over the semester. I would also like to thank my parents and friends, who motivated me to do the project and provided me with moral support, especially during the tough times of the pandemic.

Debajyoti Dasgupta

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Why Smart Grid theme was chosen?

A smart grid is an electrical grid which includes a variety of operation and energy measures including Advanced metering infrastructure (of which smart meters are a generic name for any utility side device even if it is more capable e.g. a fiber optic router), Smart distribution boards and circuit breakers integrated with home control and demand response (behind the meter from utility perspective), Load control switches and smart appliances, often financed by efficiency gains on municipal programs (e.g. PACE financing), Renewable energy resources, including capacity to charge parked (electric vehicle) batteries or larger arrays of batteries recycled from these, or other energy storage, Energy efficient resources, Sufficient utility grade fiber broadband to connect and monitor the above, with wireless as backup. Sufficient spare if "dark" capacity to ensure fail over, often leased for revenue.

Electronic power conditioning and control of the production and distribution of electricity are important aspects of the smart grid.Smart grid policy is organized in Europe as Smart Grid European Technology Platform.Roll-out of smart grid technology also implies a fundamental re-engineering of the electricity services industry, although typical usage of the term is focused on the technical infrastructure. The

idea of A "smart electricity system" has moved from conceptual to operational in the last few years. The smart grid has undergone significant innovation, with demand response being one other important focus areas. Further, the advanced metering infrastructure (AMI) has made it possible to collect the usage data and to communicate with other AMI devices.

1.1.2 Power Control and information flow

- This ability to control usage is called demand-side management (DSM)
- Research showed that, it could translate into as much as $59 billion in societal benefits by 2019

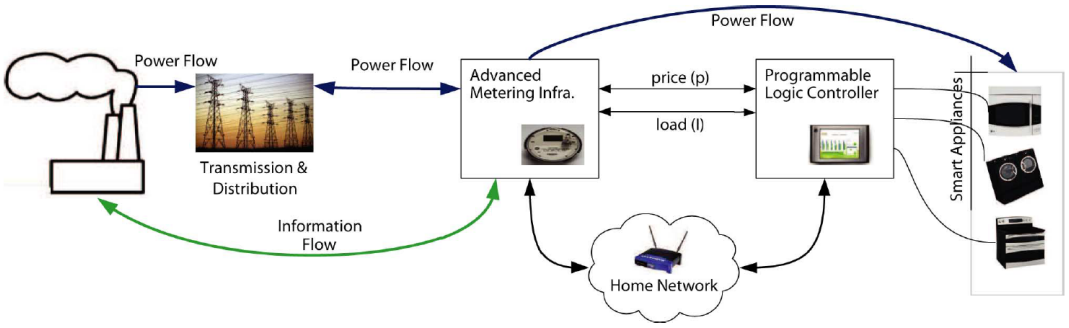


FIGURE 1.1: Power Control and information flow Diagram

1.1.3 Where does optimization fit in?

Improvements in energy efficiency have not been significant enough to counteract the increasing demand Agency 2018. To overcome this situation, the deployment of intelligent devices and communication infrastructure in smart grids becomes an important initiative. By doing so, the demand side is able to play an active role in energy management to balance demand and supply. More specifically, the demand side can change the consumption profile based on information (such as electricity price and generation capacity) from the supply side. For example, the authors in Palensky and Dietrich 2011 proposed several concepts for scheduling the consumption of household appliances to reduce the electricity cost.

1.1.4 Why consider multi-agents?

While moving to the real world looking through the eyes of optimization of only myself not only is fair nor does it promote the efficient utilization of the natural resources. Moreover it is pretty common now-a-days that people usually live in tall towers and huge buildings which may have hundreds of apartments within them, each having their own machines and using their own quota of electricity ignorant of what the consumption levels of the neighbour might be. But we want to study here, if we can make intelligent systems that model behaviour of the neighbours can there be steps that can be taken by the owners to optimize the overall power usage by the entire building which may lead to sustainable utilization of the resources.

1.2 Objective and Work Done

The major issue with using Smart Grids as the only source of control on the energy utilization is that many of the second world and third world countries are yet to get fully functional smart grid set up and working for mass usage. But most of the people in these countries do own a mobile phone or a laptop with good internet connection. So we propose a two way modeling in this thesis report.

1. The model we propose can be used for giving output a schedule of devices that user will needed during the day given the jobs that need to be scheduled. There can be two different modes of scheduling in this case, a dynamic scheduling and a static scheduling. In dynamic scheduling the user can give any job on any machine during the day, and even increase or decrease the number of available machines or cancel the ongoing jobs. The static scheduling is the case when the jobs are given in advance that needs to be done and the user will not add any other job during the day. This can be thought of as a special case of the dynamic scheduling. The main dependence of this functionality is the manual use, because the manual user may be delayed in scheduling the job on the device at the time mentioned by the schedule in the output. This is taken care in the model we propose during training, as it will take input the state of the environment which is simulated such that it is very close to a human behaviour. The user will be using an application that will be using the model to give a scheduling and update the application after every time a new job is

scheduled on a device as per the schedule. If delayed input, the application will generate alternate scheduling for rest of the jobs that are left. The major benefit is that this proposal can be used in any household even without AMI.

2. The second way of using the model is along with the smart Grid technology. The model will be receiving continuous input from the advanced metering infrastructure. Further the user may also install automatic scheduling of job facility and sensors to detect whether a job has been scheduled on a particular device or not. this is a complete automation of the process of completing all the household jobs without the user needing to interfere in any format. If the user wants then they can interrupt the scheduling and add new jobs or devices, similar to the formulation of the dynamic scheduling in the above point.

We will be utilising the Deep Reinforcement Learning approach as described in Mnih et al. 2015 to build a robust and scalable model that will be able to learn how to schedule jobs efficiently to get minimum cost with time of the day rating system and penalty for over utilising the energy quota. We have proposed new modelling of the environment for the Reinforcement Learning Agent and efficient way to represent constrains for resource scheduling on household devices by taking motivation from Mao et al. 2016.

- Apart from the above, we have also additionally improvised newer methods of training. Our first new method is termed as incremental training. Incremental training basically trains the model by selectively training the layers on the sub task of the problem one by one. An easy example would be, if we want to train an agent to learn playing a football game with 11 players, we first teach the model to learn to play with one player, then with two players and so on up to eleven players. We studied how incremental training benefits the training of the DQN model both on the fronts of performance and time taken for the training. We ave achieved a performance boost of **15.38%** in case of complex models like ResNet by exploring more observation space within less number of training steps.

- Extending this training we show that coarse training with 15 minute time steps and fine tuning with 5 minute time can give a performance boost of **16.67%** even in case of efficiently trained model like InceptionNet. Also we show this this approach provides a better scheduling which covers mode of the day.

- The second improvement in training is the use of a predictor network. A predictor network is a model that has learnt some method of optimization earlier, and we use this network to train another network so that the performance improves further. We show that using a predictor network in our model to train an LSTNet as a feature extractor, there is an preformance improvement of **6.8%** than without using a predictor.

- We additionally propose an efficient mathematical formulation for modeling the satisfaction of the human user with a given scedule based on the deadline and the requirement and show how well the agent trained with above mentioned algorithms perform as compare to the greedy FCFS strategy which is much near to human behaviour.

- We then extend this modelling to include multiple agents working together in a cooperative environment and compare the performance of models like QMIX Rashid et al. 2018 and VDN Sunehag et al. 2017 to competitive baseline heuristics using binary search and ,mixed integer programming optimizations.

1.3 Layout of Thesis

In this section of Chapter 1 of the thesis report it was mainly described what was the motivation behind choosing the topic of the thesis and how can the modeling that we will propose actually benefit the society. In the following Chapter 2 we will start by describing the problem statement that we aim to solve in this thesis. In chapter 3 we will be giving a brief review of the literature review from the papers in which related work were published and the motivations that can be taken from those papers for solving our problem. Chapter 4 will be giving a detailed description of the inputs that we will be receiving for the modelling purpose and the output that we will be returning to the user. It also describes the mathematics behind the modelling and brief description of the actual implementation of the model. The results and observations are discussed in Chapter 5 of the thesis report. Finally we conclude the report with discussion on the future work in Chapter 6 that can be done continuing the work that has been proposed in this thesis. At the end we add a section of Bibliography adding the literature referred in this thesis.

Chapter 2

Problem Statement

In this problem will be focusing on scheduling the appliances that are used by the user (or the client). Subsequently, many research papers have suggested several mechanisms to schedule the consumption of the appliances. The authors in Maharjan et al. 2013, Maharjan et al. 2016 applied game theory to model the interaction between the utility companies and the customers to reduce the power consumption. A real-time pricing scheme was adopted in Zhao et al. 2013, and a genetic algorithm was utilized to minimize the electricity cost. Instead of scheduling the appliances in the residential area, the scheduling of industrial loads was considered in Gholian A. 2016. Then, an incentive scheme was applied in Ehsanfar and Heydari 2018 to encourage more households to participate in load scheduling.

Since the client side devices are in the control of the user we have decided to apply scheduling techniques to this field first. Since there can be a wide variety of appliances that user can use in everyday life, like TV, refrigerator, Air Conditioners etc. we have decided to focus on selected appliances. For selecting this appliance we will firs t categorize the appliances into some major groups. The classes that we categorize the devices into are :

1. Appliances that need to finish batch task within deadline and have stages defined in the process
    - Eg. Washing Machine and Dish Washer
2. Appliances that have to run in periods (Periodic Appliances)
    - Eg. Air Conditioner and Refrigerator. (they stop running when a certain temperature is achieved and start running again later)
3. Appliances that have to perform the same job multiple times(sometimes with different settings)

- Eg. Microwave and Toaster.(Many Food Scheduling type)

Our major assumption will also involve the fact that we will be using Time of The Day pricing. Also we will be including the Demand Response based penalties, that is crossing a certain limit set will include penalties on the user side.

The devices are classified into different groups because of their functioning. The functioning of the devices in different classes is described as follows:

1. The devices in the first category have well defined processes that it needs to complete to finish a batch of jobs. Between these processes the machine can be preempted and we can continue the process later on. For example in washing machine, the process to clean clothes involve - wash, rinse and spin. so after washing we can schedule some other appliance and then come back to rinsing.

2. The devices in the second category usually run in periods. They try to achieve dome target and once the target is achieved, it stops working, later and later on starts working again. For example, in the case of air conditioner, the device works up until the temperature set bu the user is not reached. Once the temperature in the settings is reached, the thermostat senses that and the air conditioner is automatically switched off. Later when the temperature of the room changes, the ac will start working again. Similar is the principle followed by the refrigerator also. Since these devices usually work on a range of temperature we will set a limit to the operable conditions, like in the range of $T_0 \pm R$ where $R \leq 0.01$ the AC may turn off, where T_0 is the target and R is cushion range.

3. Finally in the third type of devices, they are usually tasked with doing the same job multiple times, sometimes just with different settings. For example in case of Microwave oven we need to heat a lot many foods usually with different settings for the heat at which the microwave should operate. Also while heating a food, we cannot preempt the process, otherwise it will render the process useless.

An interesting observation here is that, if we consider the entire process of washing as a single process then the washing machine can be classified into the Group 3 devices also. This is because the washing machine will also be washing different set of clothes with different settings, like cotton, jeans, mild rinsing etc.

2.1 Objectives we aim to achieve

Our main objective in this problem will to reduce the peak demand on the client side and spread it equivalently among the entire duration. This is mainly because first of all if the peak demand is higher than the limit then there may be severe penalties. Secondly if the peak demand is higher especially during the time of the day when the cost of supplied electricity is high, it will cost the the user very much and is also not efficient management of energy. Also since we are using time of the day pricing as well as penalties for crossing the limit, our objective function will also include minimization of the client cost (demand side).

Second while we are minimizing the cost of the client it is important to keep the user satisfied also. Suppose if the microwave was scheduled to heat a food at 8 a.m. in the morning, but the scheduler schedules the heating of the food for 9 p.m., then the user will be very unsatisfied with the service of the Reinforcement Learning agent. Another situation will be if we provide to much cushion for Refrigerator (say 3) then the food may get spoilt and the user will be again be very unsatisfied. So we will also be keeping track of the satisfaction score and the task of our objective function will be to maximize the satisfaction score.

Thus, in short our objective will be

1. Minimize Demand side (client) cost
2. Reduce peak Demand
3. Maximize the satisfaction score

2.2 Input for the problem

Since we will be making use of Reinforcement learning, the input should contain mainly of three important things - the environment, the reward that was provided when the output action is taken in the previous step. The following are the description of what will be included for each these inputs :

1. The **environment** mainly will contain a set of parameters that will describe the state of the device at the current point of time. The devices that belong to each of the three groups (as described in Section 1) will have different types of states, like the washing machine needs to mention the state/process in which it is currently running and the time remaining to complete the running process, where as the refrigerator may provide its current temperature

or the difference between target and current temperature and the microwave may in addition add the mode which it has to be set to operate in currently. We may also include how much of each of the job is completed till now. The environment of the system may contain any other details also that will provide any extra information about the state of the device. The environment of the device is very important since it will help in the further decision making.

2. The **reward** will be basically the the output of the reward function that will be calculated by the reinforcement learning agent in the previous time stamp. This reward will also influence the decision making in the future. So the Reinforcement Learning agent is trying to maximize the future reward. So if the reward in the previous step was very low it needs to take some decision accordingly to increase the future reward. For this the reward function should be carefully determined.

We can further extend the approach by storing the Experience Replay. also if we are using ϵ-greedy strategy, then we will need the value of ϵ in the previous step to implement ϵ-decay.

In addition to the above inputs we will also need to provide a list of inputs that need to be scheduled. This list may be dynamic or a static list (while testing), though for training purpose we will use a static list of the jobs. The Job description will contain the type of jobs, setting it needs and the device on which he job needs to schedule.

Also the Reinforcement learning agent should have the information of the working of the device (if any). The working of the device basically means what type of setting requires how much time, say if it is configured that the rinsing of the washing machine will take 15 minutes(say). Also this list will contain which process should necessarily come after which other process, like it is necessary that spin comes after rinse in washing machine. Sometimes switching between the different states of the device also requires time and loss/gain of energy (to be compensated from the source), so details about that will be helpful in the input (if modelled properly). These information should be available to the Reinforcement Learning agent in advance.

2.3 Output for a Good Scheduling

Since we are working with Reinforcement Learning, the main component of the output will be the action that was taken by the Reinforcement Learning agent based on some policy and the Reward that was achieved after taking that action. Following is the description of the above outputs

1. The **action** taken will contain maninly of the following parameters

   (a) Which device was selected (scheduled) to operate in the current time slot. This will basically be a pointer to the device or a list of devices that need to be activated and the jobs will be scheduled on these devices.

   (b) How much time slice was allocated to the process to be scheduled currently. This time slice will basically decide how much time the device scheduled will run the selected job for, Once the device completes the time slice before completing the job, the process will be preempted from it, whereas if it finishes earlier than the time slice then it will willingly give the rest slice for scheduling the next appliance.

   (c) which job is done on the current scheduling.That is from the list of jobs which are still pending to be completed, it gives a pointer to the job that needs to be scheduled now as scheduling this device will be beneficial.

   (d) Which phase (in the list of process to be done, like the rinse phase in case of washing machine) will the device scheduled will be executing the job/process in. This will mainly be the next phase that needs to be done as we have the job completed until the previous phase.

   According to the job that was scheduled, the time allotted to the job may vary (that is the time slice will not be constant). Also if a device scheduled take very low power then next device will be scheduled simultaneously. This will help in reducing the peak demand and help in utilizing the power supplied efficiently also.

2. The **reward** will simply refer to the reward that will be earned after taking the current action as specified in the output. The reward will be computed using the reward function which will be trained using the deep Q-network. The reward function will contain a combination of all the objectives that we want to achieve, that is minimizing the demand side cost, reducing the peak demand and maximizing the satisfaction of the user. There will be mainly two kinds of reward function, the episodic reward and the continuous reward. It is needed to be decided as to which type of reward function should be adopted for the model. This will actually depend on the implementation of the model that we adopt.

In addition to the above details, the output may also contain how much of the currently scheduled job will be completed in the current time slice of the device that is provided in the output. This is important because it is not necessary that the job will be completed as it will be preempted when the time slice expires.

2.4 Extending to Multi-Agent Systems

The multi-agent problem statement is basically dealing with handling multiple apartments in a single building with each apartment having their own agents to solve their scheduling and trying to optimize their rewards (primarily the bill amount). extend the problem statement to the domain of multi agent system we modify the above mentioned constraints in the following way.

2.4.1 Disclosure of information

Since the owners may be either be very open or be scared about sharing their data,our study will be extending to different levels of information sharing. Primarily were are concerned about the knowing the actions of the other agents in our neighbourhood and what state they are in. Based on this the information that is shared may be from PUBLIC, SHARED or PRIVATE (no sharing at all) knowledge set. We will primarily be trying to optimize the case for no knowledge sharing during the execution time and try to improve on the results by slowly increasing the amount of knowledge that is shared.

2.4.2 Cooperation

For the initial study we assume that we are implementing our agents in a friendly neighbourhood, were everyone is ready to compromise for other. Hence we first make study on cooperative environment and look into, how well can we optimize, if all the agents were to fully cooperate with one another.

2.4.3 Input and Output

The input for the multi agent environment, will be very much similar to the one described above for a single agent scenario, with the modifications to handle multiple apartments in a single building, each with their won intelligent agent.

- We are considering N houses, each house will have a single RL Agent (hence in total N RL agents will be acting together).
- Each house will have three category of devices (as mentioned above), with zero or multiple instances of each category of device. Primarily house H_i will be having
  - MA_i devices of category 1 type

  - MB_i devices of category 2 type
  - MC_i devices of category 3 type

- For the ease of modelling our observation space and to make the model flexible for different amounts of machines in each house, we will be specifying an upper limit to the number of devices of each category that can be added to a particular house. Thus MA_i, MB_i and MC_i will belong to the mentioned limit.

- A primary assumption to make the study of the complex domain a bit simpler, we will currently be assuming that the average energy profile of the devices of a particular category remains same across all houses. This means that is washing machines are being used in different houses, then all the washing machines are the same model, belonging to the same company. Assuming same energy profile is a realistic assumption since a large number of families use 2-star or 3-star rated devices, which have near about same energy consumption during the run time.

- Houses will also need to declare the number of jobs they want each machine to handle during the day. That is house H_i will be having

  - JA_i jobs to be scheduled on category 1 type devices
  - JB_i jobs to be scheduled on category 2 type devices
  - JC_i jobs to be scheduled on category 3 type devices

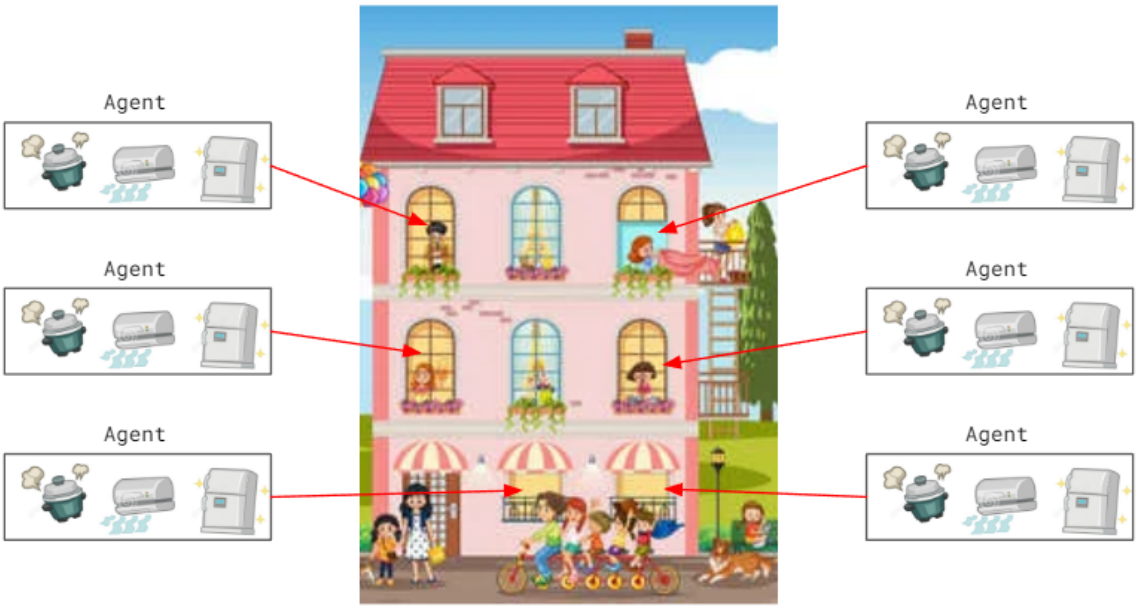


FIGURE 2.1: Multi Agent Environment

Chapter 3

Literature Review

3.1 Deep Q Networks (DQN)

- Paper - Human-level control through deep reinforcement learning - Deep Mind - Mnih et al. 2015
- Introduces the concept of building a Reinforcement Learning model by learning the Q-function (Action-value function) by using a Deep Neural Network to predict the function and hence the policy that will help to decide the action that needs to be taken on the input state.

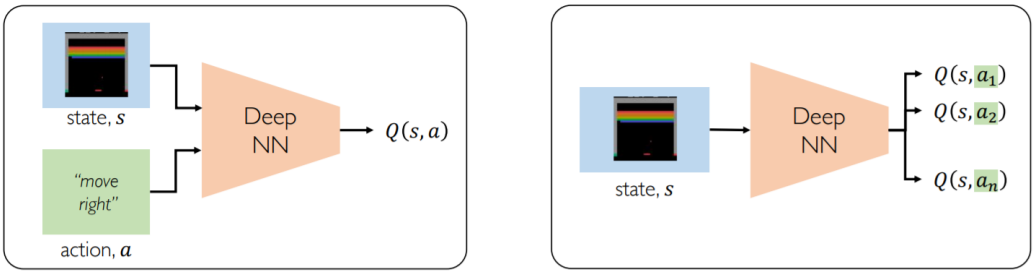


FIGURE 3.1: Using Deep NN to model Q-function (Mnih et al. 2015)

- **Accomplishment**
  - The paper shows huge success on games involving complex decision making process and many interlinking states. These games were mainly from the Atari 2600

  - It was also shown that if we were to randomly choose between experience replay and new experiences using a tuned parameter ϵ, then the performance can be further improved.

- **Complexity:**
  - Can model scenarios where the action space is discrete and small
  - Cannot handle continuous action spaces
- **Flexibility:**
  - Cannot learn stochastic policies since policy is deterministically computed from the Q function
- We will be using Deep Q-network because it's shortcomings will not affect us.
  - The states in the application will be discrete - jobs that need to be completed
  - Decisions need to be taken at discrete times, when a job or batch of jobs are completed.

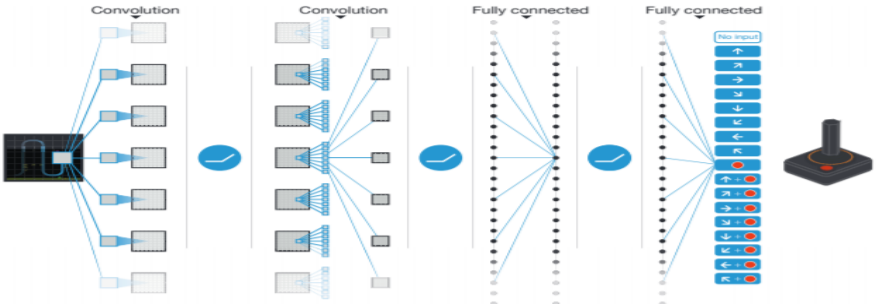


FIGURE 3.2: CNN Architecture (Mnih et al. 2015)

- From the results we see that the on the games which had too many hidden states to explore the model did not perform better than a human being. However in most other games the DQN model was able to outperform even the most professional players.

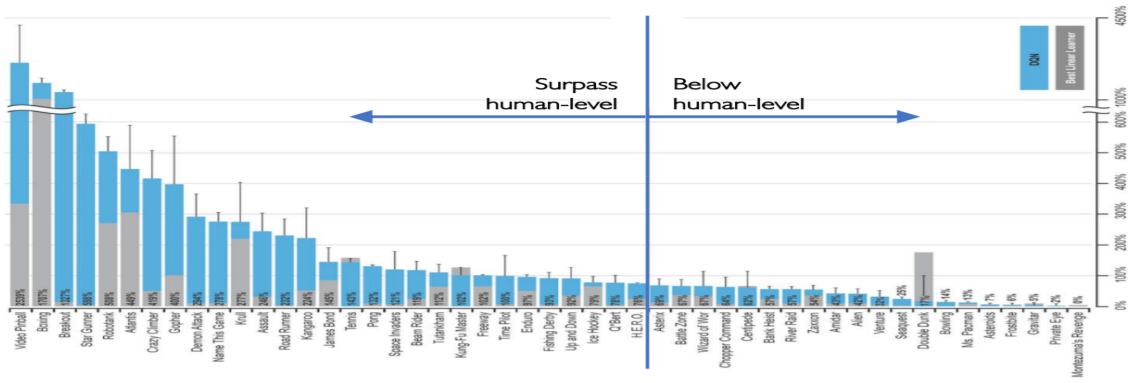


FIGURE 3.3: Comparison of the DQN agent with the best reinforcement learning methods in the literature

3.2 Epsilon-Greedy Action Selection

Epsilon-Greedy is a simple method to balance exploration and exploitation by choosing between exploration and exploitation randomly.The epsilon-greedy, where epsilon refers to the probability of choosing to explore, exploits most of the time with a small chance of exploring.Mnih et al. 2015

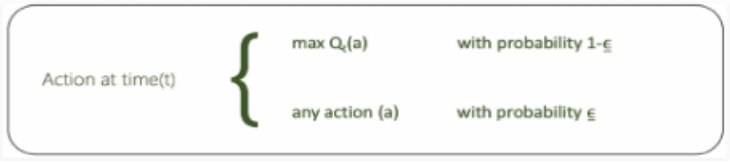


FIGURE 3.4: Epsilon Greedy Logic

3.3 RL Agent for managing power demand between grid and battery

Mary Mammen and Kumar 2019

1. RL agent to manage the operation of storage devices in a household and is designed to maximize demand-side cost savings
2. The proposed technique is data-driven, and the RL agent learns from scratch how to efficiently use the energy storage device given variable tariff structures

3. This paper also brings into light two new models for solving some of the issues in DQN (which will be discussed later)

4. **Accomplishment**

   (a) The paper shows that the model was quite successful even with only Time of the Day pricing, and had saving of 6-8%

   (b) Savings of 12-14% can be obtained if the utility follows the ToD pricing along with rewards from the DR (Demand Response) program

5. **Observations**

   (a) Agent performs better on high capacity batteries than low capacity batteries

   (b) This is because the agents trained on the low capacity batteries have not seen states that are experienced by the high capacity batteries

   (c) But the high capacity batteries have seen the states that are seen by the low capacity batteries

   (d) Hence, the Deep Q Network has better performance when it can explore more number of states

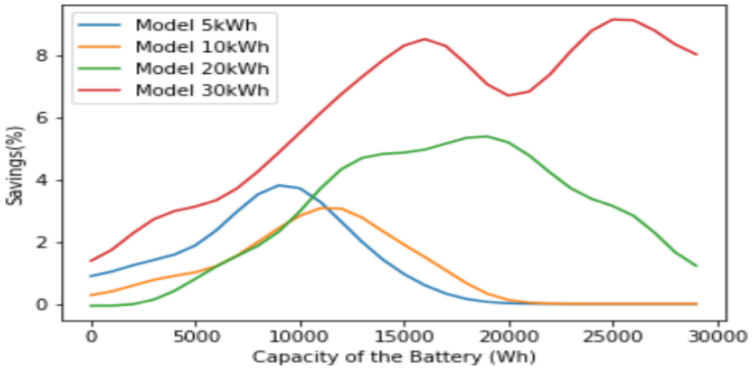


FIGURE 3.5: Storage Capacity vs Cost Saving

3.4 Appliance Scheduling Optimization in Smart Home Networks

Qayyum et al. 2015

1. this paper proposes a solution to the problem of scheduling of a smart home appliance operation in a given time range

2. In addition to power-consuming appliances, this paper also adopt a photo-voltaic (PV) panel as a power-producing appliance that acts as a micro-grid

3. An appliance operation is modeled in terms of uninterruptible sequence phases, given in a load demand profile.

4. Goal is - minimizing electricity cost fulfilling duration, energy requirement, and user preference constraints

5. An optimization algorithm, which can provide a schedule for smart home appliance usage, is proposed based on the mixed-integer programming technique.

6. **Results**

   (a) Simulation results demonstrate the utility of the proposed solution for appliance scheduling.

   (b) The paper further show that adding a PV system in the home results in the reduction of electricity bills and

   (c) the export of energy to the national grid in times when solar energy production is more than the demand of the home

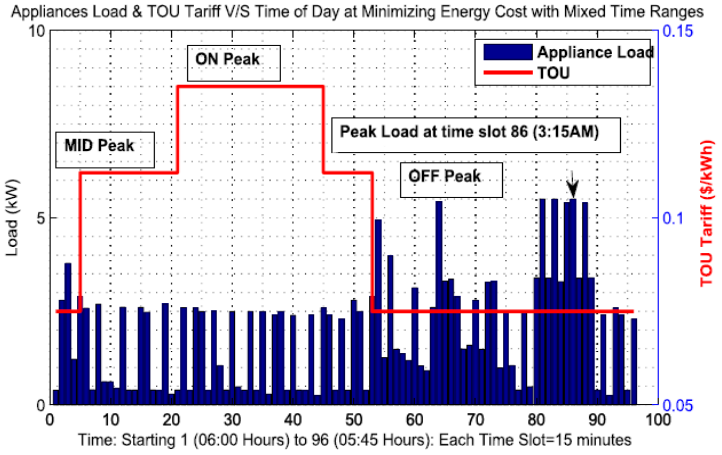


FIGURE 3.6: Load pattern of appliances operating on mixed time range.

3.5 A Distributed Algorithm

Chavali, Yang, and Nehorai 2014

1. Paper - A Distributed Algorithm of Appliance Scheduling for Home Energy Management System

2. Demand side management encourages the users in a smart grid to shift their electricity consumption in response to varying electricity prices

3. This paper, we propose a distributed framework for the demand response based on cost minimization.

4. Each user in the system will find an optimal start time and operating mode for the appliances in response to the varying electricity prices.

5. In order for the users to coordinate with each other, we introduce a penalty term in the cost function, which penalizes large changes in the scheduling between successive iterations.

6. **Results** Numerical simulations show that the optimization method that is proposed in the paper will result in

   (a) Lower cost for the consumers

   (b) lower generation costs for the utility companies

   (c) lower peak load

   (d) lower load fluctuations.

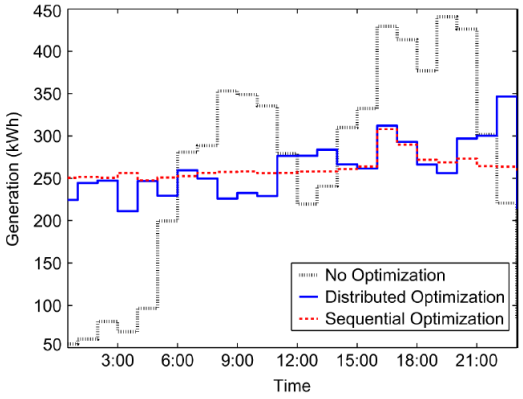


FIGURE 3.7: Comparison of hourly utility company generation.

3.6 Value-Decomposition Networks For Cooperative Multi-Agent Learning

Sunehag et al. 2017

- This paper studies the problem of cooperative multi-agent reinforcement learning with a single joint reward signal. This class of learning problems is difficult because of the often large combined action and observation spaces. In the fully

centralized and decentralized approaches, it was found that the problem of spurious rewards and a phenomenon we call the "lazy agent" problem, which arises due to partial observability.

- These problems are adressed in this paper by training individual agents with a novel value decomposition network architecture, which learns to decompose the team value function into agent-wise value functions. This work proposes a way to have separate action-value functions for multiple agents and learn them by just one shared team reward signal. The joint action-value function is a linear summation of all action-value functions of all agents. Actually, by using a single shared reward signal, it tries to learn decomposed value functions for each agent and use it for decentralised execution.

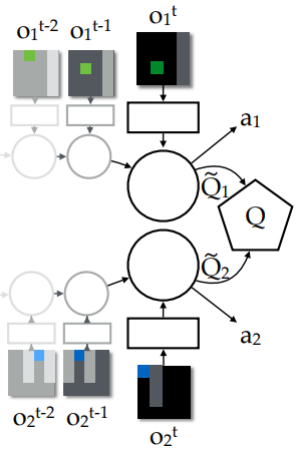


FIGURE 3.8: Value-decomposition individual architecture showing how local observations enter the networks of two agents over time (three steps shown), pass through the low-level linear layer to the recurrent layer, and then a dueling layer produces individual "values" that are summed to a joint Q-function for training, while actions are produced independently from the individual outputs

- For 2 agents case the reward will be $r(s,a) = r_1(o^1, a^1) + r_2(o^2, a^2)$. Then the total Q function is:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi]$$

$$= \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_1(o_t^1, a_t^1) | \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi] + \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_2(o_t^2, a_t^2) | \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a}; \pi]$$

$$=: \bar{Q}_1^{\pi}(\mathbf{s}, \mathbf{a}) + \bar{Q}_2^{\pi}(\mathbf{s}, \mathbf{a})$$

- Experimental evaluations are performed across a range of partially-observable multi-agent domains and show that learning such value-decompositions leads to superior results, in particular when combined with weight sharing, role information and information channels.

3.7 QMIX: Monotonic Value Function Factorization for Deep Multi-agent RL

Rashid et al. 2018

For most of the multi-agent reinforcement learning experiments in the laboratory, extra state information can actually be provided during centralized training. However, a efficient strategy that can properly extract decentralized policies doesn't exist yet. Therefore, the authors of this paper proposed a novel value-based method that can train decentralized policies in a centralized end-to-end fashion, which is called QMIX. QMIX estimates total joint action values as a complex non-linear combination of per-agent values conditioned only on local observations. This strategy structurally enforces that the joint-action value function is monotonic in nature in the per-agent values.

- A fully cooperative multi-agent task can be seen as a Dec-POMDP. At each time step, each agent chooses an action based on their current observation and the history information. After applying the action, each agent would produce the corresponding Q-value.
- Although training is centralized, execution is decentralized, i.e., the learning algorithm has access to all local action-observation histories τ and global states, but each agent's learned policy can condition only on its own action-observation history.
- The main idea of QMIX is that the authors think that the full factorization of VDN is not crucial in order to be able to extract decentralized policies that are fully consistent with their centralized counterparts. Actually, we only need to make sure that the argmax joint action performed on total Q yields the same result as a set of individual argmax operations performed on every agent. We usually define this perspective as Individual-Global-Max(IGM):

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}$$

- With the above mentioned assumption, every agent can participate in a decentralized execution solely by choosing greedy actions with respect to its Q. Monotonicity can be enforced through a constraint on the relationship between total Q and individual Q:

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \epsilon A$$

- For every agent, this paper follows the same network architecture mentioned in the VDN, which is a recurrent-based network. In order to fulfill the constraint, QMIX uses an architecture consisting of agent networks, a mixing network, and a set of hyper networks. The following diagram shows the detail of the QMIX:

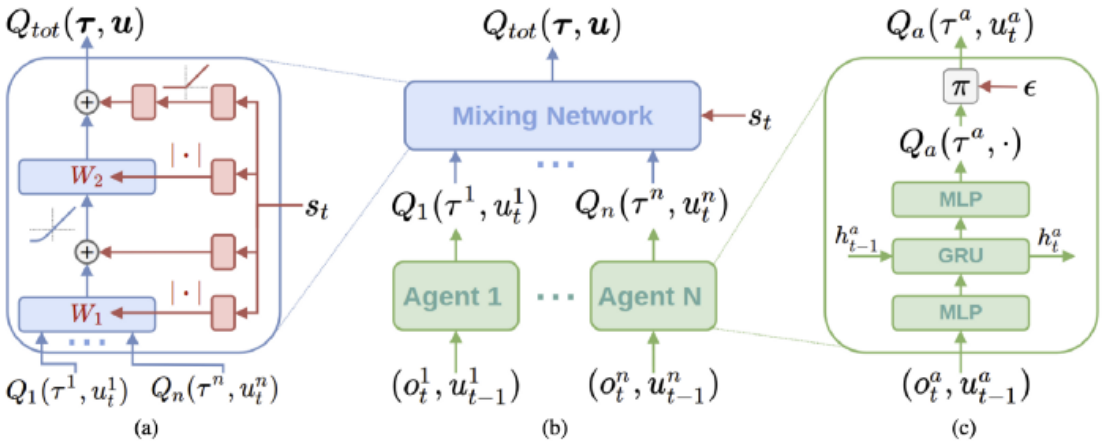


FIGURE 3.9: (a) The architecture of mixing network (b) QMIX overview (c) The architecture of agent network

- The main contribution of QMIX is the mixing network. The mixing network is a feed-forward network that outputs the total Q value. It inputs the individual Q value for each agent and mixes them monotonically. In order to follow the monotonic constraint, the hyper-networks of QMIX use the global state as its input and output the weight of the mixing network. And each hyper-network consists of a single linear layer, followed by an absolute activation function so that it can follow the constraint.

- In this paper, the environment of the experiment is StarCraft II micromanagement problems, which is also called SMAC. In this paper, the environment of the experiment is StarCraft II micromanagement problems, which is also called SMAC.

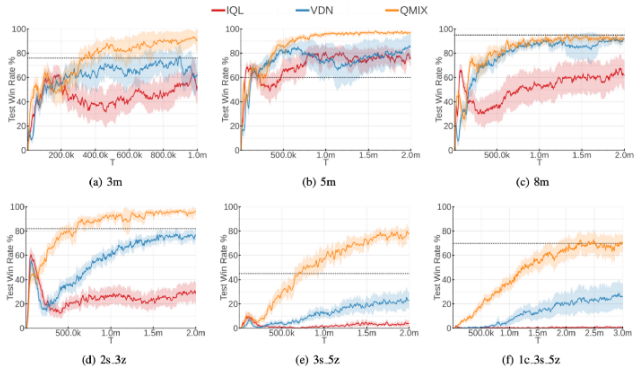
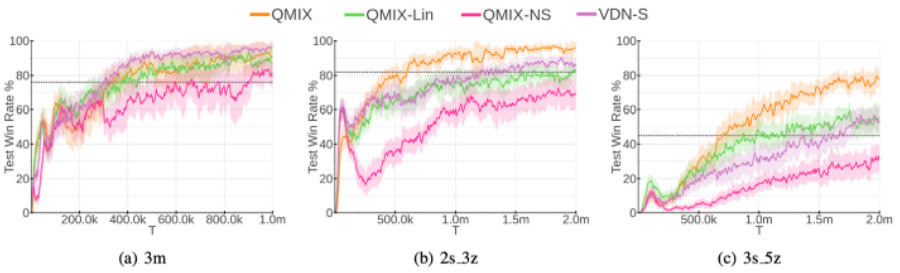


FIGURE 3.10: Win rates for IQL, VDN, and QMIX on six different combat maps. The performance of the heuristic-based algorithm is shown as a dashed line.

In this paper, they also proposed three different ablations in order to show the advantage of QMIX:

- First, they analyze the significance of extra state information on the mixing network by comparing it against QMIX without hyper-networks. Thus, the weights and biases of the mixing network are learned in the standard way, without conditioning on the state. They refer to this method as QMIX-NS.
- Second, they investigate the necessity of non-linear mixing by removing the hidden layer of the mixing network. This method can be thought of as an extension of VDN that uses the state s to perform a weighted sum over Qa values. They call this method QMIX-Lin.
- Third, they investigate the significance of utilizing the state in comparison to the non-linear mixing. To do this they extend VDN by adding a state-dependent term to the sum of the agent's Q-Values. This state-dependent term is produced by a network with a single hidden layer of 32 units and a ReLU non-linearity, taking in the state s as input (the same as the hyper-network producing the final bias in QMIX). They refer to this method as VDN-S.



3.8 A Genetic Algorithm Based Power Consumption Scheduling in Smart Grid Buildings

- With the growing adoption of smart grid technologies and the widespread distribution of smart meters, it has become possible to monitor electricity usage in real time within modern smart building environments. Utility companies are now implementing different electricity prices for each time slot, taking into consideration peak usage times. This paper introduces a novel power consumption scheduling algorithm for smart buildings that utilize smart meters and real-time electricity pricing.

- The proposed algorithm Lee and Bahn 2014 dynamically adjusts the power mode of each electrical device in response to changes in electricity prices. Specifically, we model the electricity usage scheduling problem as a real-time task scheduling problem and demonstrate that it is a complex search problem with exponential time complexity. To address this challenge, our scheme employs an efficient heuristic based on genetic algorithms, which significantly reduces the search space and identifies a reasonable schedule within an acceptable time budget.

- Experimental results under various building conditions reveal that the proposed algorithm effectively reduces the electricity costs for a smart building. On average, the algorithm achieves a 25.6% reduction in electricity charges, with a maximum reduction of up to 33.4%. This demonstrates the potential of our approach to optimize power consumption in smart building environments while accounting for real-time electricity pricing, ultimately leading to significant cost savings and improved energy efficiency.

Figure 3.11 shows an example of power consumption scheduling for each electricity demand as time goes on. In the figure, peak time represents the time period at

which the price of electricity becomes high. As shown in the figure, it is desirable to change the state of each device to a low-power mode during peak time.

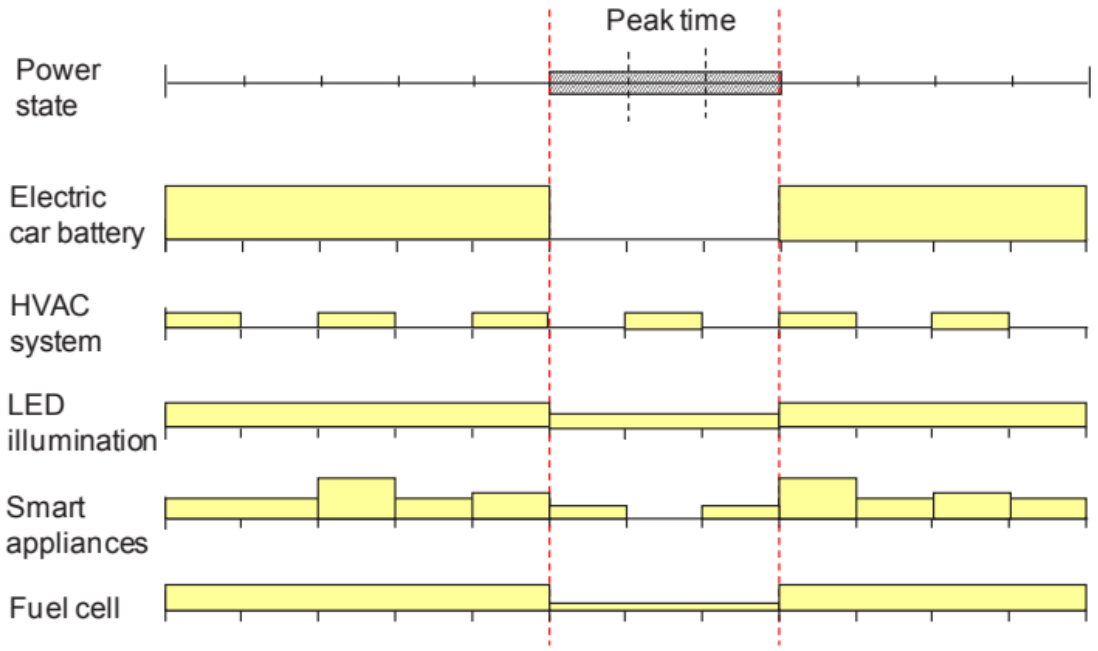


FIGURE 3.11: An example of power consumption schedulig

3.8.1 Proposed soltuion

This paper cuts down the huge search space of the genetic algorithm using an evolutionary computation method to find an approximated schedule within a reasonable time budget. Specifically, a genetic algorithm, which is a probabilistic search method based on natural selection and population genetics, is used. Figure 3.12 depicts a brief flow of the genetic algorithm. A certain number of schedules are initially generated and they form the initial set of schedules. Among schedules in the set, two schedules are selected and then merged as one scheduled by the crossover and mutation operations, and then the schedule set is evolved by replacing a schedule in the old set with the newly generated schedule. This process is repeated until the schedule set converges.

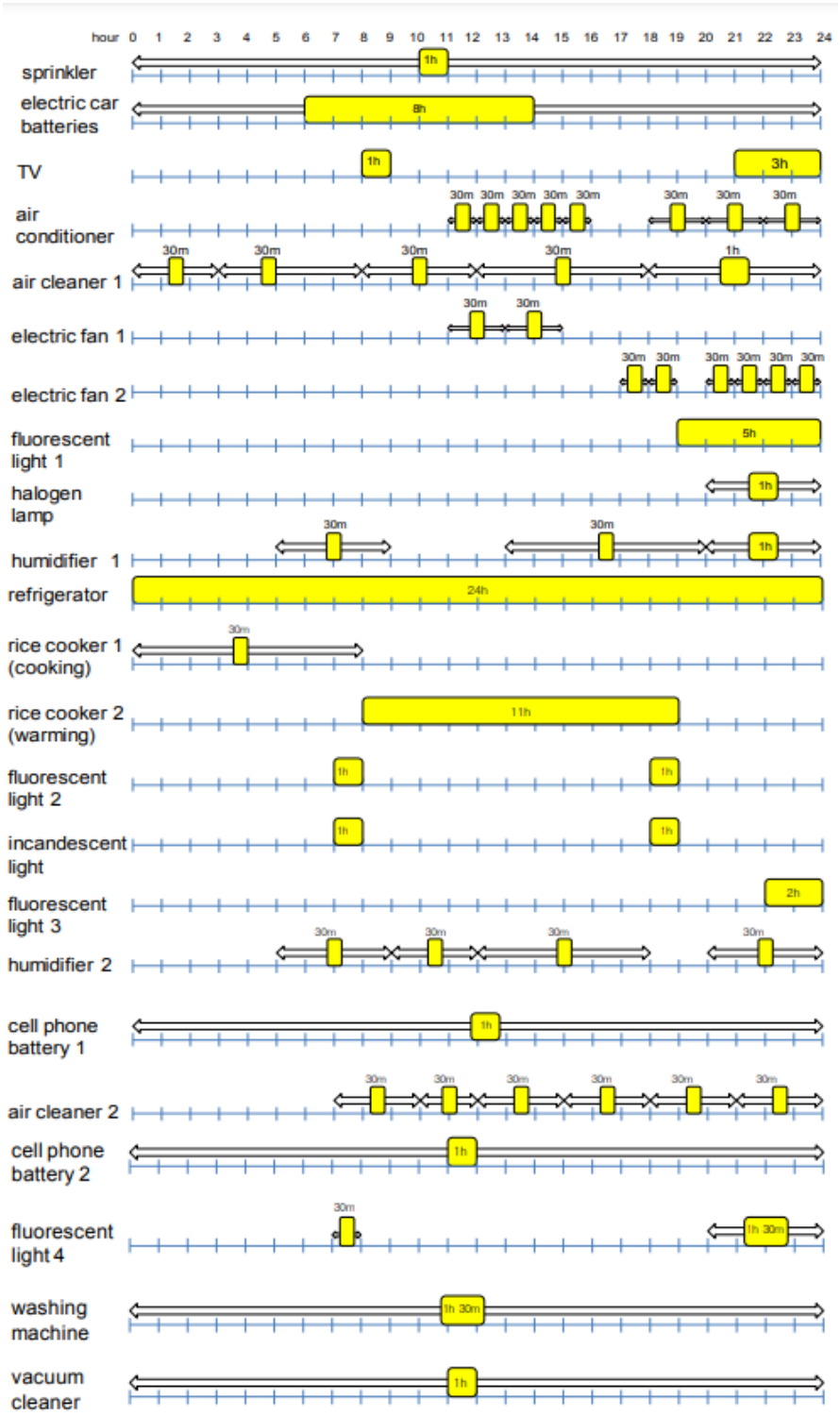


FIGURE 3.12: An example of electricity demands and their scheduling slots. An example of electricity demands and their scheduling slot

3.8.1.1 Encoding and Constructing Initial Scheduling Se

The proposed algorithm represents a schedule using a matrix, with rows corresponding to electric devices and columns representing the 24 hours of a day. Each matrix entry is either 0 or 1, where 0 indicates the device is off and 1 signifies it is on. For devices with multiple power modes, each mode is treated as a separate row in the matrix. This simplifies the encoding compared to using multiple values for each entry.

Devices have different power demand behavior over time, and scheduling can be classified as interruptible or non-interruptible based on the flexibility in splitting tasks across time slots. The algorithm maps interruptible tasks to multiple non-interruptible tasks with finer-grained slots. The scheduler initially generates a set of 1000 random schedules.

3.8.2 Selection Operation

A selection operation chooses two schedules in the current scheduling set that will be the parents of the next generation. This paper applies the most common practice, which gives the best schedule four times more probability of being selected than the worst one [3]. The value of each schedule is evaluated by the electric charge of the schedule. The normalized value of a schedule is defined as follows.

$$NV_i = (R_w – R_i) + (R_w – R_b)$$

Where Rw, R_b, and R_i are the ranks of the worst schedule, the best schedule, and schedule i, respectively, in the current generation. Based on the normalized value, a roulette wheel selection is applied as a selection operator. It assigns each schedule i a slot with a size equal to its normalized value NV_i. The pointer of the wheel is a random real number ranging from 0 to NV_i. A schedule whose slot spans the

pointer is selected and becomes a parent. Note that this is based on a probabilistic rule that favors better schedules but is not deterministic. We use a probabilistic approach here to avoid too much discrimination. That is, if we deterministically select the best schedules, they may dominate the scheduling set rapidly, potentially causing premature convergence to a local optimum.

3.8.3 Crossover and Mutation Operation

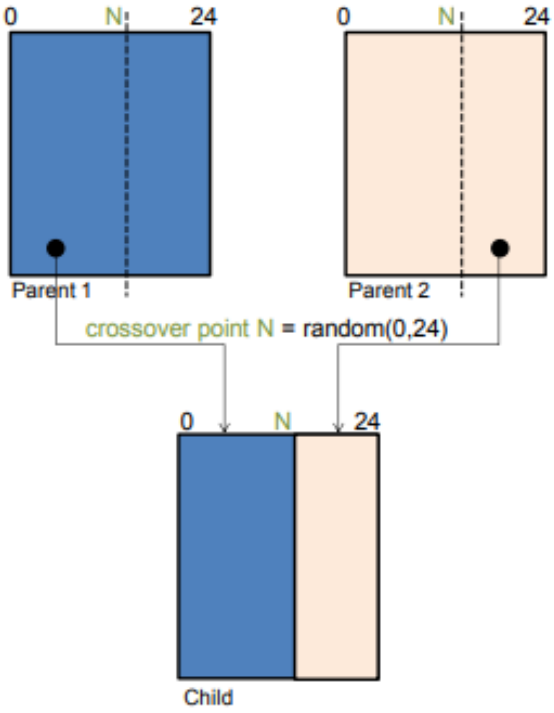


FIGURE 3.13: Crossover operations in the proposed scheduling algorithm

After selecting the parents, a crossover operation is executed. The goal of the crossover is to combine beneficial segments from the parents to create a child that leads to improved schedules over time. We employ a traditional one-point crossover technique, which generates a random crossover point and then swaps segments between the two parents to produce offspring. In our method, the crossover point is generated by cutting a matrix column, as depicted in Figure3.13.

This paper does not explicitly define mutation operations, which alter the generated child to maintain diversity, as the crossover operations should incorporate adjustment routines that serve the purpose of mutation. Specifically, when applying the classical one-point crossover, an infeasible schedule might be created, meaning that the scheduling sequence does not meet the time constraints for each device. Such a situation arises when the crossover point cuts the sliding window of a scheduling unit. As a result, we adjust the operation so that the child primarily inherits the scheduling unit from parent 1, while the remaining slots are completed by parent 2. If any empty slots remain, they are randomly filled by selecting any location within the sliding window

3.8.4 Replacement Operations

Upon creating a child schedule, the new generation is formed by replacing a schedule from the current generation with the newly generated child. In this paper, the schedule with the highest electricity charge in the current generation is replaced by the new child schedule, which is the most frequently employed replacement operation.

3.8.5 Results

Experiments were performed with six different scale buildings, where the electricity demands are 50, 100, 150, 200, 250, and 300, respectively. Before showing the comparison results, Fig.3.14 plots the electricity charges of the best and the worst schedules in the current search space and their average as the generation progresses. As shown in the figure, the quality of schedules improves significantly according to the evolution of the generation and, finally, the convergence. As shown in the figure, the proposed scheduling system performs the best irrespective of the building scale. The performance gain of GA-based scheduling is 27.0% on average and up to 36.4% compared to the original scheduling system. As shown in the figure, the

proposed scheduling system performs the best irrespective of the building scale. The performance gain of GA-based scheduling is 27.0% on average and up to 36.4

Figure 3.16 shows the electricity charges of the proposed GA-based scheduling system and the original system that does not use it. In the original system, each scheduling unit is randomly located among the release time and the deadline. For comparison purposes, the Greedy scheduler is also simulated. The Greedy scheduler locates each scheduling unit in the time slot in which the electricity price is lowest.

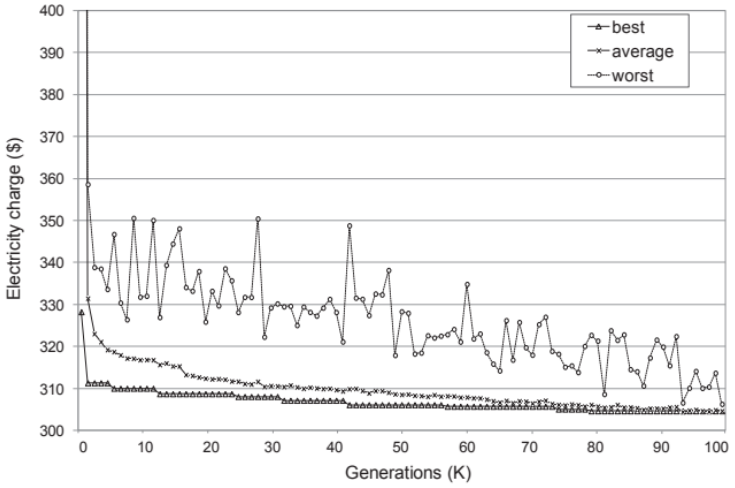


FIGURE 3.14: convergence of the scheduling set as time progresses

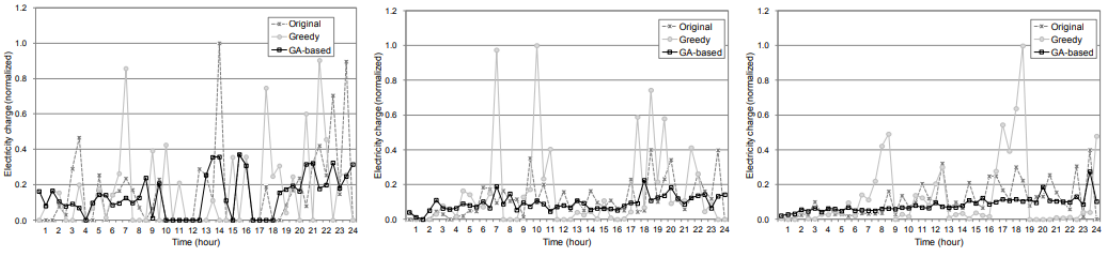


FIGURE 3.15: Electricity charges of each time slot of a day

For the exact analysis of this interesting result, the electric charge of each time slot of a day is also examined. Figure 3.15 plots the normalized electricity charge of the three schedulers as time progresses within a day. As shown in the figure, the

proposed GA-based scheduling system exhibits relatively uniform electricity charges irrespective of the time zone. The reason behind such balancing is that it finds a schedule that does not exceed the threshold of the progressive stage system at all time slots, including non-peak zones as well through collaborative scheduling.

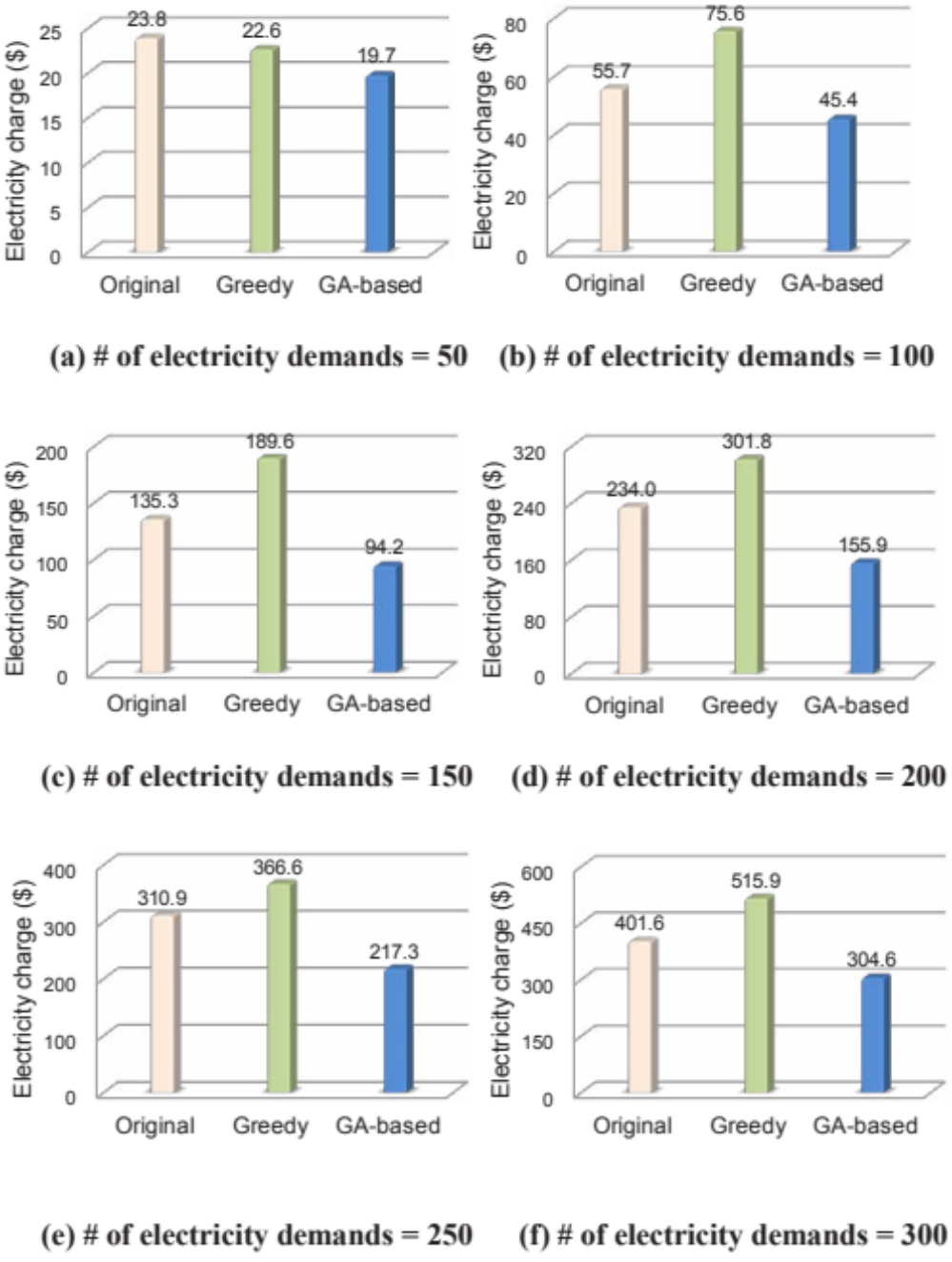


FIGURE 3.16: Comparison of electricity charges

Chapter 4

Problem Formulation and Methodogy

4.1 About the Devices

Since in the limited amount of time we cannot include all the household devices that are used everyday, we try to classify the devices in 3 broad categories. In addition to this, the model we built is a robust model and will be highly scalable so that given the device and its specification.

1. **Category-1**
   (a) Appliances that need to finish batch task within deadline and have stages defined in the process Eg. Washing Machine and Dishwasher
   (b) The devices in this first category have well defined processes that it needs to complete a batch of jobs. Between these processes the machine can be preempted and continue the later. For example in washing machine, the process to clean clothes involve - wash, rinse and spin. so after washing we can schedule some other appliance and then come back to rinsing.

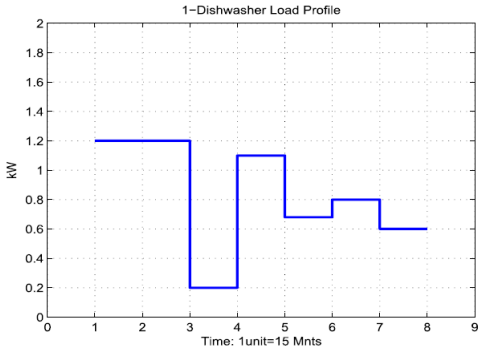


FIGURE 4.1: load profile for Dish Washer

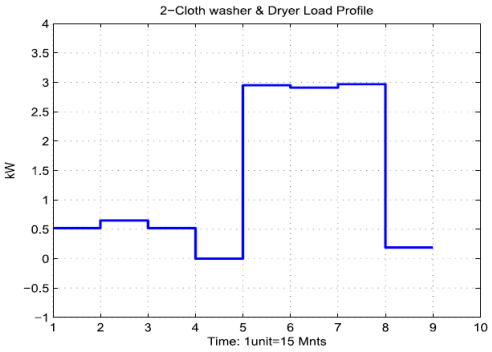


FIGURE 4.2: load profile for Washing Machine

2. **Category-2**

   (a) Appliances that have to run in periods (Periodic Appliances) Eg. Air Conditioner and Refrigerator. (they stop running when a certain temperature is achieved and start running again later)

   (b) They try to achieve some target and once the target is achieved, it stops working, later and later on starts working again.

   (c) For example, in the case of an air conditioner, the device works up until the temperature set by the user is not reached. Once the temperature in the settings is reached, the thermostat senses that, and the air conditioner is automatically switched off.

   (d) Since these devices usually work on a range of temperature we will set a limit to the operable conditions, like in the range of $T_0 \pm R$ where $R \leq 0.01$ the AC may turn off, where T_0 is the target and R is cushion range.

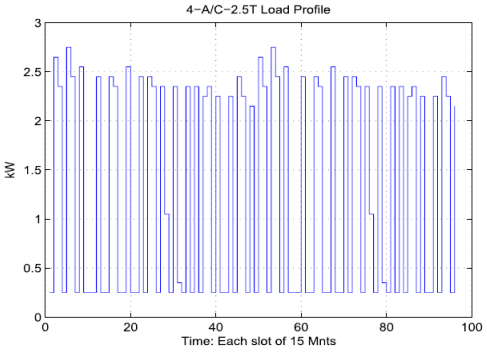


FIGURE 4.3: load profile for refrigerator

3. **Category-3**

(a) Appliances that have to perform the same job multiple times (sometimes with different settings)

(b) Eg. Microwave Oven and Toaster. (Many Food Scheduling types)

(c) Devices in this category same job multiple times, sometimes just with different settings.

(d) For example in the case of a Microwave oven we need to heat a lot many foods usually with different settings for the heat at which the microwave should operate. Also while heating a food, we cannot preempt the process, otherwise, it will render the process useless.

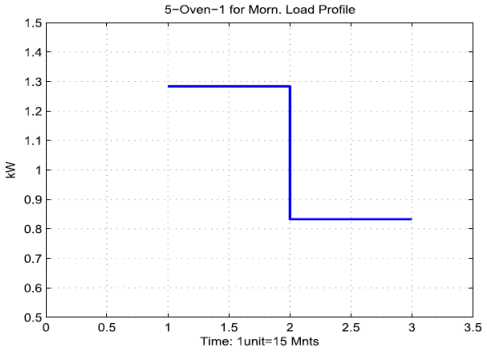


FIGURE 4.4: load profile for the oven in morning

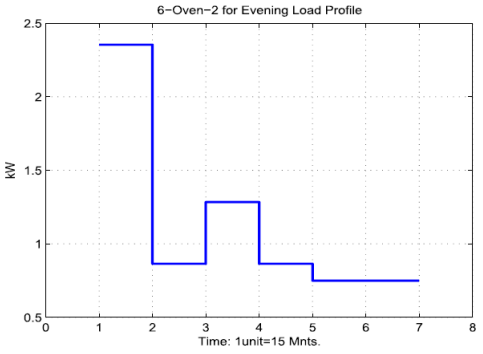


FIGURE 4.5: load profile for the oven at night

Some Additional Points

- An interesting observation here is that, if we consider the entire process of washing as a single process then the washing machine can be classified into the Group 3 devices also.
- This is because the washing machine will also be washing different sets of clothes with different settings, like cotton, jeans, mild rinsing, etc
- An interesting observation here is that, if we consider the entire process of washing as a single process then the washing machine can be classified into the Group 3 devices also.
- This is because the washing machine will also be washing different sets of clothes with different settings, like cotton, jeans, mild rinsing, etc

Symbol	Definition
i	Index of appliance to be scheduled
k	Time slot over a given period of a day
j	Index of number of load phases associated within each appliance
n_i	Set of number of un-interruptible load phases associated with each appliance i
N	Set of number of appliances for scheduling
m	Maximum number of time slots available in a day
P_{ij}^k	Load variable assigned to an appliance i, having load phase j during time slot k
C^k	Tariff in dollars in the time slot k
X_{ij}^k	A binary decision variable with value 1, if ith appliance with load phase j in the time slot k is processed; otherwise 0.

FIGURE 4.6: Symbols used in problem formulation

4.2 Objective Functions

4.2.1 Reduce peak Demand

Qayyum et al. 2015

1. Our main objective in this problem will be to reduce the peak demand on the client side and spread it equivalently over the entire duration.
2. This is mainly because first of all if the peak demand is higher than the limit then there may be severe penalties.
3. Secondly if the peak demand is higher, especially during the time of the day when the cost of supplied electricity is high, it will cost the user very much and is also not efficient management of energy.

$$f_L = \min_X \sum_{k=1}^{m} \sum_{i=1}^{N} \sum_{j=1}^{n_i} (P_{ij}^k X_{ij}^k - q)^2 \tag{4.1}$$

q is the average load of appliances considered

$$q = \frac{\frac{1}{4}(\sum_{i=1}^{N} \sum_{j=1}^{n_i}) P_{ij}}{24} \tag{4.2}$$

4.2.2 Minimize Demand side (client) cost

Mary Mammen and Kumar 2019 Also since we are using time of the day pricing as well as penalties for crossing the limit, our objective function will also include minimization of the client cost (demand side). Power utilized multiplied by the cost, X is a vector of $X_{i,j}^k$

$$f_c = \min_x \sum_{k=1}^{m} C^k (\sum_{i=1}^{n} \sum_{j=1}^{n_i} P_{ij}^k X_{ij}^k) \tag{4.3}$$

4.2.3 Maximize the satisfaction score

while we are minimizing the cost of the client it is important to keep the user satisfied also. Suppose if the microwave was scheduled to heat food at 8 a.m. in the morning, but the scheduler schedules the heating of the food for 9 p.m., then the user will be very unsatisfied with the service of the Reinforcement Learning agent.
User Satisfaction is a term that varies from person to person. But we create a mathematical measure of the same to create an estimate (as per the use of the device). All the devices will have user satisfaction scores on two fronts, one is the delay in the start of the job scheduled on the device which is defined as the difference between the time of job submission and the time of scheduling. Following is the user satisfaction for each device:-

1. For the first category of devices (eg. washing machine) satisfaction will also be measured on the fact as how much time is taken for the job completion defined as the difference between the scheduling of the job and the completion of the job.

2. For the second category of devices (Eg. Air Conditioner), these devices will be running continuously whenever needed. The other metric of the satisfaction score of these devices will be the deviation from the range that it is required to operate in, defined as the difference between the current temperature and the nearest endpoint of the optimal operating range.

3. The metric for the third category of devices namely microwave ovens, is same as that of the first category devices, that is the time taken for job completion

4.3 Additional Goals

In addition to the above goals, we will also be deploying the model in some cloud-based systems with some specified API endpoints. Also, this can be implemented into a mobile app or a web app. Using the above API, people will be able to make

use of the model built to implement static scheduling of jobs in countries such as India. Also countries such as Singapore which have an advanced metering scheme in use will be able to use full-fledged dynamic scheduling of devices

4.4 Observation Space

- The observation space is represented as an image of shape (number of machines × number of jobs × number of distinct machines). A number of distinct machines in our case remains fixed at 3 as we have proposed classification into 3 classes for all household machines.
- This method of representation of observation space has taken motivation from Mao et al. 2016. The benefit of this representation are as follows
  ‘

  1. Each layer will deal with scheduling jobs of a particular type of machine. Say layer 1 of the image deals with the washing machine, then it will deal with scheduling washing clothes-related jobs. Hence the redundant states (such as scheduling jobs oven jobs on the washing machine) are reduced significantly, as compared to combining the 3 layers in a single image.
  2. The dimension of each layer is (number of machines X number of jobs) for a particular type layer, hence the image size is smaller and scalable on both machine dimension as well as job dimension.
  3. In this formulation, we do not need to burden ourselves to find a model that can learn the proper splitting of energy among the different RL agents (for different devices). A proper split will be learnt from the several CNN filter whose weights are learned during training.

The following image shows the image representation of the intermediate state in the environment. It can be interpreted as follows (assume for the ith row and jth column):

- The black cell denotes that machine(i) cannot schedule on the job(j). This can be due to several different reasons either machine(i) is already running a job different from (j), machine I have been allocated to an incomplete process, or job(j) belongs to a process that has been allocated to a machine that is different from (i).
- Light color (more towards white) means less energy used in the process, and darker cells (more towards black) imply less power utilization.

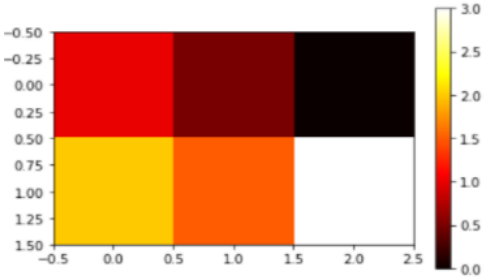


FIGURE 4.7: Observation Space single layer

4.5 Action Space

- The action space is represented using a bitmap. Since we need to consider scheduling all the subset of machines, the set bit of the bit map will represent which machines should be scheduled at the time moment. The bitmap is a binary representation, so the action space range will be equal to the range of the decimal equivalent of the binary numbers.

$$Action_{Min} = 0 \tag{4.4}$$

$$Action_{Max} = 2^{\#machine} \times 2^3 \tag{4.5}$$

4.5.1 Reduction of the redundant states

- or example, say we have 10 microwaves, 10 ACs, and 10 Washing Machines. When we take them in a single layer, the action space size will become $2^(10 + 10 + 10) = 1$ billion approx.
- However now say we use separate observation space in three separate layers, then the action space size will reduce to $2^10 + 2^10 + 2^10 = 3072$ states. We can see a significant reduction in the size of the action space in the two cases
- This can be also visualized using the following analogy. Now we have say 29 microwaves, 28 washing machines, and 28 air conditioners, then we will have $2^29 + 2^28 + 2^28 = 1$ billion states (approx) in the action space, that is using the same number of action spaces we will be able to use more machines of each type.

4.6 Model Description

4.6.1 Deep Q Network

1. Here we use Deep Neural Network to approximate the Q-Values, which would have been obtained by Mnih et al. 2015

$$Q^{new}(s_t, a_t = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma * max_a Q(s_{t+1}, a) \tag{4.6}$$

2. Loss Function: Excluding the satisfaction score

$$R(t) = -[Cost\_electricity(t) * demand(t) + penalty(demand(t))] + user\_satisfaction(t) \tag{4.7}$$

$$cost\_saving = (1 - \frac{Cost(RLAgent)}{Cost(Baseline)}) * 100 \tag{4.8}$$

3. **Fixed Q-target (Target Estimation)**Kim et al. 2019
   - Using a separate network with a fixed parameter for estimating target value
   - At every T step, we copy the parameters from DQN (Deep Q-Network) network to update the target network
   - This improves the stability of the Neural Network as updates are not made to the target function after every batch of learning

4.6.2 Prioritized Experience Replay(PER)

1. Experiences (state, action, reward, next_state) are stored and chosen based on priorities. Mary Mammen and Kumar 2019
2. Probability of being chosen for a reply is given by stochastic prioritization
   - $P(i) = \frac{p_i^a}{\sum_k p_k^a}$
   - If a=0, then it is pure randomness
   - if a=1 then select only the experiences with highest priorities.
3. Updates to the network are weighted with Importance Sampling weight
4. Importance Sampling weight $(IS) = (\frac{1}{N} * \frac{1}{P(i)})^b$
5. a and b are hyper parameters to be trained

4.6.3 Double Deep Q - Network

1. Double DQN introduced by Hado van Hasset is used to handle the problem of overestimation of the Q-values

2. Taking the maximum Q-value will be noisy during the initial phase of training if non-optimal actions are regularly given higher Q-value than the best action.

3. The solution involves using two separate Q-value estimators, each of which is used to update the other. Using these independent estimators, we can unbiased Q-value estimates of the actions selected using the opposite estimator

Basic Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Double Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma\, Q'(s_{t+1}, a) - Q(s_t, a_t))$$

estimated/expected Q-value

$$a = \max_a Q(s_{t+1}, a)$$

$$q_{estimated} = Q'(s_{t+1}, a)$$

FIGURE 4.8: Double Deep Q Network Equation

4.6.4 Dueling Double DQN

1. The value of Q(s, a) is computed as the sum of the value of being in that state V(s) and the advantage of taking action at that state A(s, a)

2. DDQN can learn which states are (or are not) valuable without having to learn the effect of each action at each state since it's also calculating V(s).

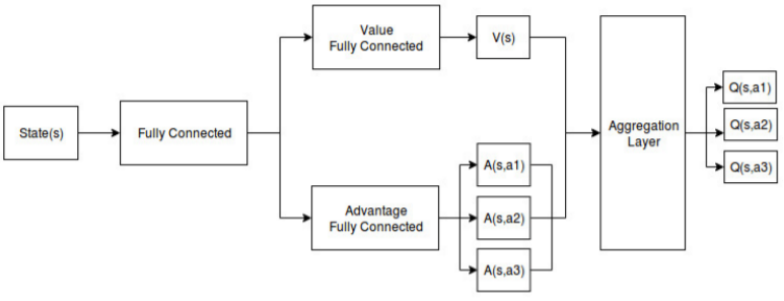


FIGURE 4.9: Dueling Double Deep Q Network Architecture

4.6.5 Neural Network Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 19, 118, 32)       896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 18, 117, 64)       8256
_________________________________________________________________
batch_normalization (BatchNo (None, 18, 117, 64)       256
_________________________________________________________________
flatten (Flatten)            (None, 134784)            0
_________________________________________________________________
dense (Dense)                (None, 256)               34504960
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 34,556,097
Trainable params: 34,555,713
Non-trainable params: 384
_________________________________________________________________
```

FIGURE 4.10: Neural Network Architecture

In the subsequent study we replace the simple convolutional layer of the neural network architecture with complex feature extractor network as shown in the diagram below. We mainly experiment with three different types of feature extractor network: InceptionNet V3 4.11, ResNet50 4.12 and a simple resnet 4.13. A simple ResNet is the one in which there are only two residual connections available. the InceptionNet is a compute efficient feature extractor, and will help us study the effect of addition of 1x1 convolutional modules (which help in reduction of the number of computations) affect the learning ability of the model as compared to the ResNet models. The two ResNets help us in the study of how increasing the residual connection affects the learning capacity of the model and it's affects on the training time.

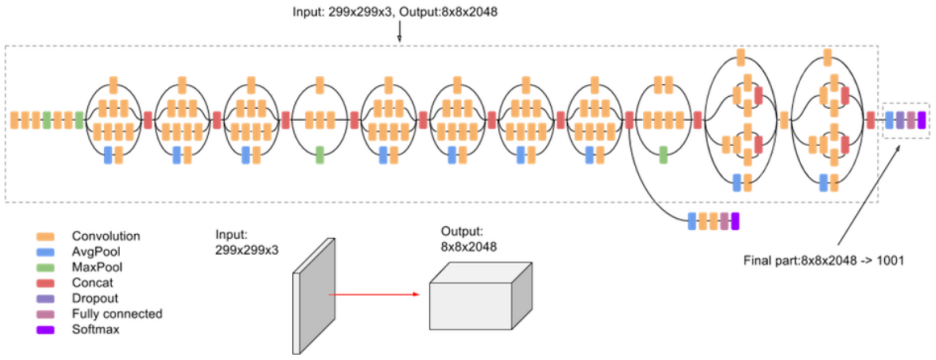


FIGURE 4.11: Inception Network V3

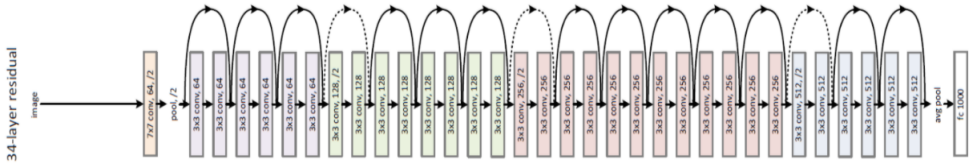


FIGURE 4.12: ResNet 50

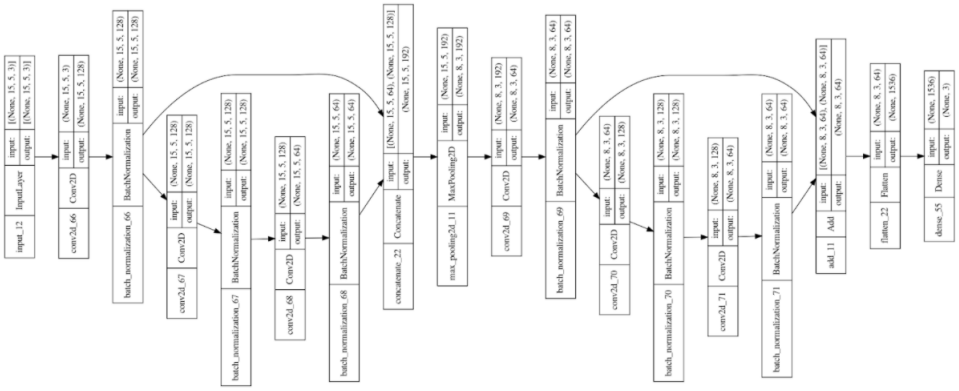


FIGURE 4.13: ResNet small

Our problem poses a unique mathematical optimization problem for getting a good schedule. So relying only on the feature extractor may not always provide the best result, so we began exploring models that also looked into the mathematical nature in the images. especially since we are dealing with time series model, it is best that we use models that make use of the recurrent connections over the time axis. Recently many such models have come up which target the mathematical nature among multivariate systems. We convert our problem into a multivariate system

by simply flattening the image matrix into a multivariate polynomial where each variable corresponds to a single sell in the image which represents the current state. So our polynomial will have (number of devices) * (number of jobs) * (number of distinct category of devices, which is three in our case) variable, where each of these variable vary over time. We chose to go with LSTNet Lai et al. 2017 model to solve this problem. LSTNet has shown prominent ability to recognize patterns in many daily use cases especially ones which have mathematical patterns in time varying format. We use this architecture to study the improvement we achieve with this architecture with/without the presence of a predictor network.

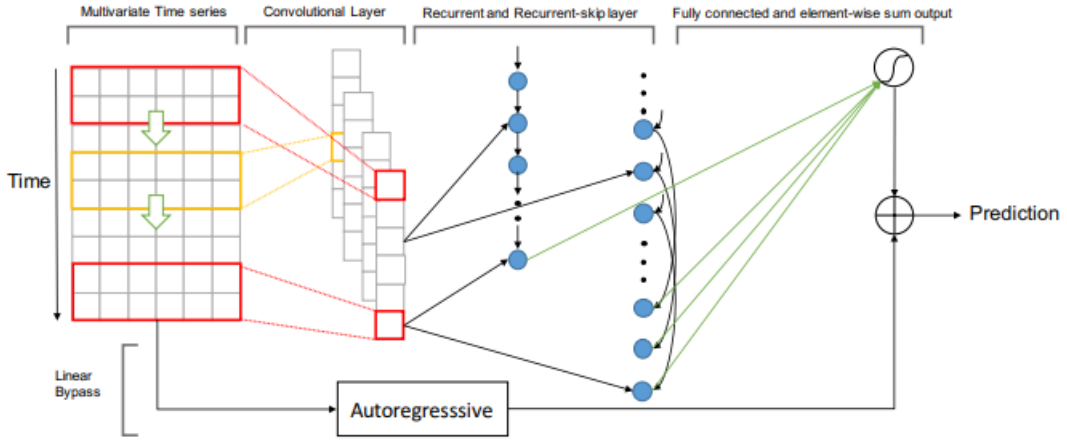


FIGURE 4.14: An overview of the LSTNet

As can be viewed from the above figure, the LSTNet 4.14 is divided into three phases, where the first phase contains the convolutional layers, which act as image feature extractor in our case. The second phase has the recurrent and the recurrent skip connections, which are very good networks capable of extracting useful information from time series data. It is here, that the essence of time series is captured by the model. We also see a presence of a Autoregressive head, which assumes that the current data depends only on the previous data and current variables. Finally all these extracted features goes through the processing of a feed forward network where the non linearity is learned to make predictions for the future time step.

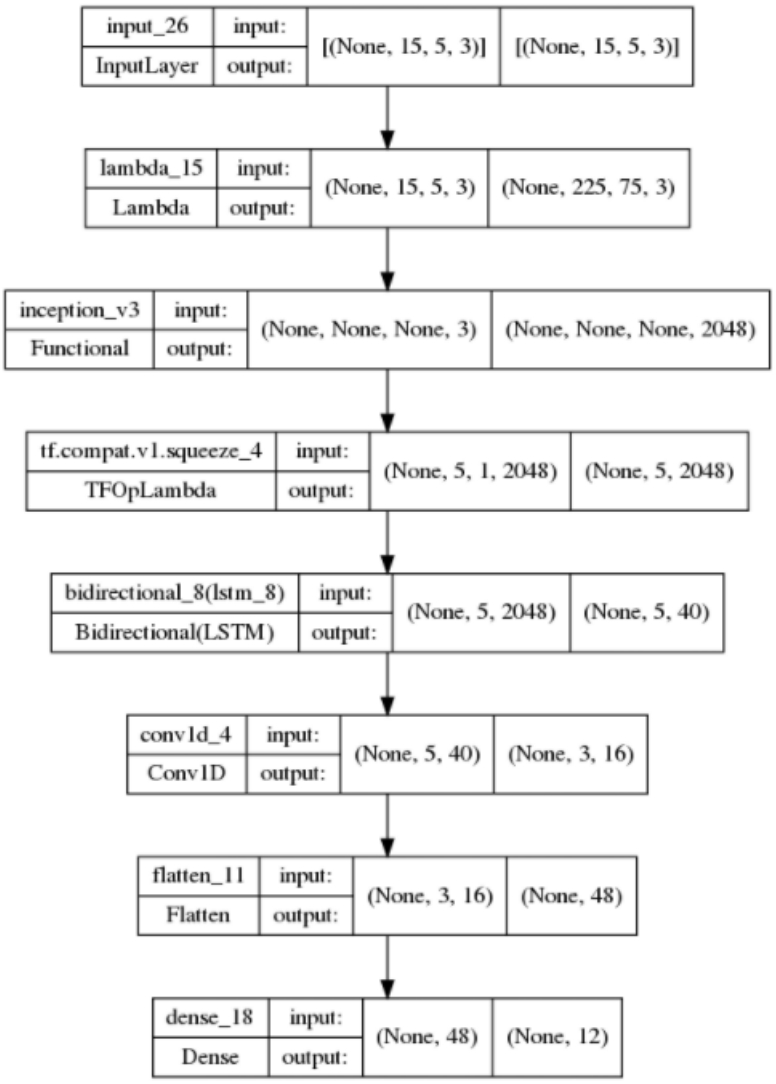


FIGURE 4.15: Appending BiLSTM to Feature Extractor

To compare the performance with the previous complex feature extractors like the Inception Net V3 4.11, we introduce a BiLSTM layer in conjunction with a 1 dimensional convolutional layer which gives the feature extractor to capture the essence of time series. We use similar layered architecture with ResNet50 4.12 and ResNet Small 4.13 as well.

4.7 Happiness

$$Happiness = p \cdot \left(\frac{g^{1.3}-75}{80} - 17\right) \cdot l_1 + \left(3.5 \cdot 10^4 \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{g}{\sigma}\right)^2}\right) \cdot l_2 + \left(\frac{(g-60)^{1.7}}{120} - 10.5\right) \cdot l_3$$

```
t = time of completion of the task (minutes)
d = deadline for completion of the task (minutes)
σ = 35 minutes (standard deviation)
g = |t-d| (completion distance)
```

$$l_1 = \begin{cases} 1 & t \leq d - 125 \\ 0 & t > d - 125 \end{cases}$$

$$l_2 = \begin{cases} 0 & t \leq d - 125 \\ 1 & d - 125 < t \leq d + 125 \text{ (Cushion Time)} \\ 0 & t > d + 125 \end{cases}$$

$$l_3 = \begin{cases} 0 & t \leq d + 125 \\ -1 & t > d + 125 \end{cases}$$

$$p = \begin{cases} +1 & \text{washing machine} \\ -1 & \text{Oven or AC} \end{cases}$$

FIGURE 4.16: Happiness Formulation

Happiness is a simple mathematical expression accounting for the satisfaction of the end user. As described in 2, we also aim to maximize the satisfaction the end user achieves from the scheduling, given that the user mentions the time of completion of a job. The three main categories that we consider our devices to be split in in the 4 have different requirements on the satisfaction and hence we customize the function as follows.

- For washing machine the user will be satisfied as soon as the clothes are washed and returned, as it gives time for drying and adding further jobs to the washing machine as well. However if the job of washing the clothes was supposed to be complete by 9AM, and gets scheduled by 9PM, the user will be very unhappy.
- For oven and Air Conditioner devices, if the user want the food to be prepared by 11AM, it is best that the machine schedules the task in the nearby time

slot as scheduling the job too early will lead to the food getting cold over due time, and scheduling too late will not serve the purpose of the job, so in either case user will be unhappy.

The values for the coefficients of the equation and the correct value for the power of the equation are extracted using grid search. The above mentioned function for happiness, is a piece-wise function, where each piece has a specific function for the happiness. The last part of the equation basically signifies that heavy penalty is imposed if the task is delayed after the deadline, so we prioritize the completion of the task within the deadline. The normal distribution near the deadline emphasises that near the deadline completion is preferred by the user. The first part of the equation is however task dependent as described in the points above. For washing machine, earlier the task is completed, the happier user is, hence it has positive weight. However, for oven and air conditioner we need to push the task to the scheduled time as doing the task too early or too late will render the job useless.

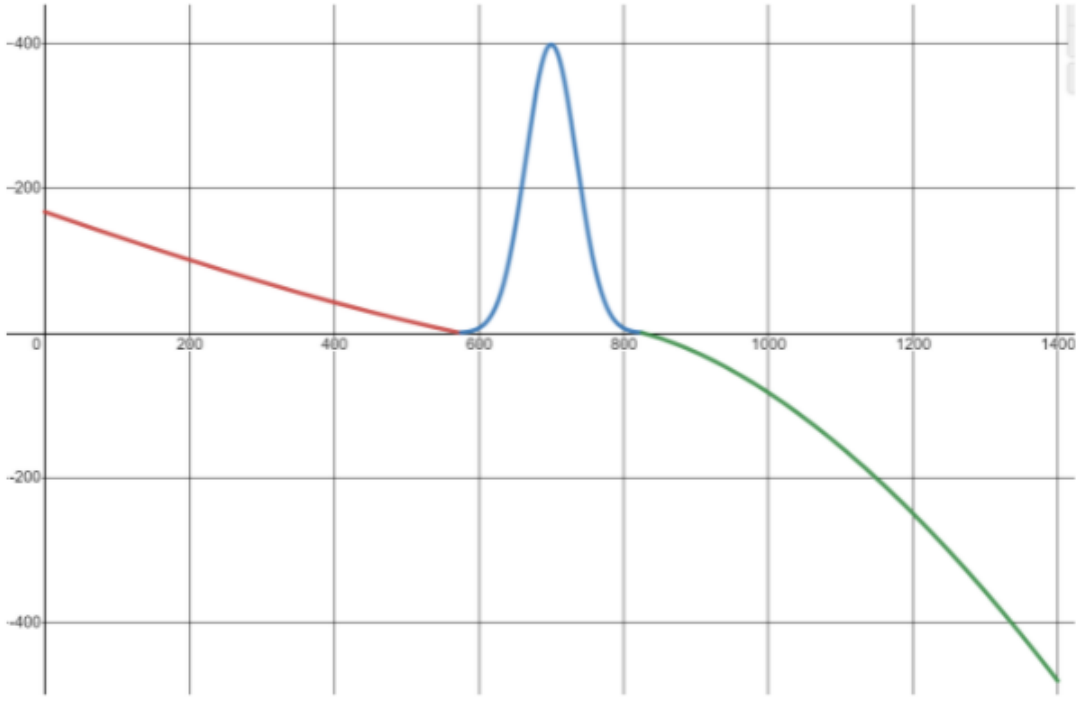


FIGURE 4.17: Happiness Formula Plot

4.8 Multi Agent Framework

4.8.1 Restructuring formulations

4.8.1.1 Objective Function

In addition to the optimization of the objective function of the individual houses by the agents, the agents must also co-operate to optimize the overall objective of the building too. That is for equations 4.1, 4.2 and 4.3, there will be an additional summation over all houses ($\sum_{H=1}^{h}$) term. This basically indicates that the total peak power and total bill amount incurred by the entire building must also be reduced while performing local optimizations too.

4.8.1.2 Observation Space

Since the jobs on the same category devices are identical (i.e. they go through the same cycles, and have the same average energy consumption in each cycle), hence we apriori allocate each of the jobs on all the available machines of the respective category by equally distributing them. For eg., say if we have 5 washing of cloth tasks to be done on 3 washing machines, we allocate 2 tasks each to the first two machines and 1 task to the third machine.

- The disadvantage of this is that, we are restricting the scheduling of tasks withing different machines. Like, if the job i_1 is scheduled on machine M_a, which is busy at the moment, it will keep waiting on the same machine, even though some other machine M_b of the same category is free.
- The major advantage of this formulation is that we don not need to limit the maximum jobs the user can schedule during the day, and hence the model so developed will be able to scale efficiently on the job domain unconstrained. Also the effect of inter-machine job scheduling is minimal since usually, average houses have a limited number of devices of the same category in the same house.

- The three factors that will now primarily affect the scheduling are:
  - **Energy consumption** on next schedule
  - **Number of jobs** that are left to be completed on the particular machine
  - **Deadline** on the upcoming job
- So accordingly, we change the new formulation of the observation space for each agent, which can now be represented as a combination of 3 vectors, each having the length same as the total number of devices available in the house, and each vector signifying one of the above-mentioned features. Hence the dimension of the observation space $(N_{machines}, 3)$. It is clearly visible from the dimensions dependency on the number of jobs, hence no limit to be provided to the user.

4.8.1.3 Action Space

- Since the jobs are now evenly distributed among the different machines, we do not need to take care of intra-device scheduling. The only decision that needs to be taken is whether to schedule a job at the current moment or not.
- So now the action space for each agent is a boolean vector of length $(N_{machines})$. This is highly scalable with a growing number of machines in the household since there will be no exponential factor to be handled anymore.

4.8.2 Technologies Used

4.8.2.1 PettingZoo

- Petting Zoo is a Python library for conducting research in multi-agent reinforcement learning, akin to a multi-agent version of Gymnasium (currently maintained version of the OpenAI-Gym).
- Actively maintained by Farama Foundation and is recognized as a standard environment implementation in lot of other libraries. The environment has

been referred to in several other related works and the developers keep it up to date with the latest works in the Multi-Agent RL industry.

- Efficiently extends gym environment. Hence the implementation from the single agent RL remains much the same with only changes for adding the multi-agent objectives and rewards.

4.8.2.2 Mava

Mava is a library for building multi-agent reinforcement learning (MARL) systems. Mava provides useful components, abstractions, utilities, and tools for MARL and allows for simple scaling for multi-process system training and execution while providing a high level of flexibility and composability. We have primarily used MAVA https://id-mava.readthedocs.io/ during the experiments described in a later chapter for its QMIX and VDN implementations and smooth support for TensorFlow.

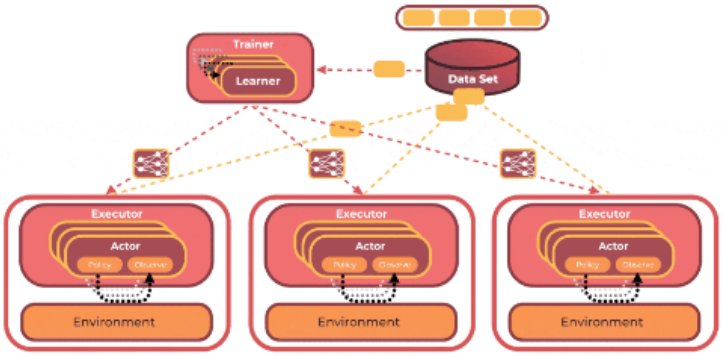


FIGURE 4.18: Internal working flow of MAVA

4.9 Genetic Algorithm

4.9.1 Genetic Space

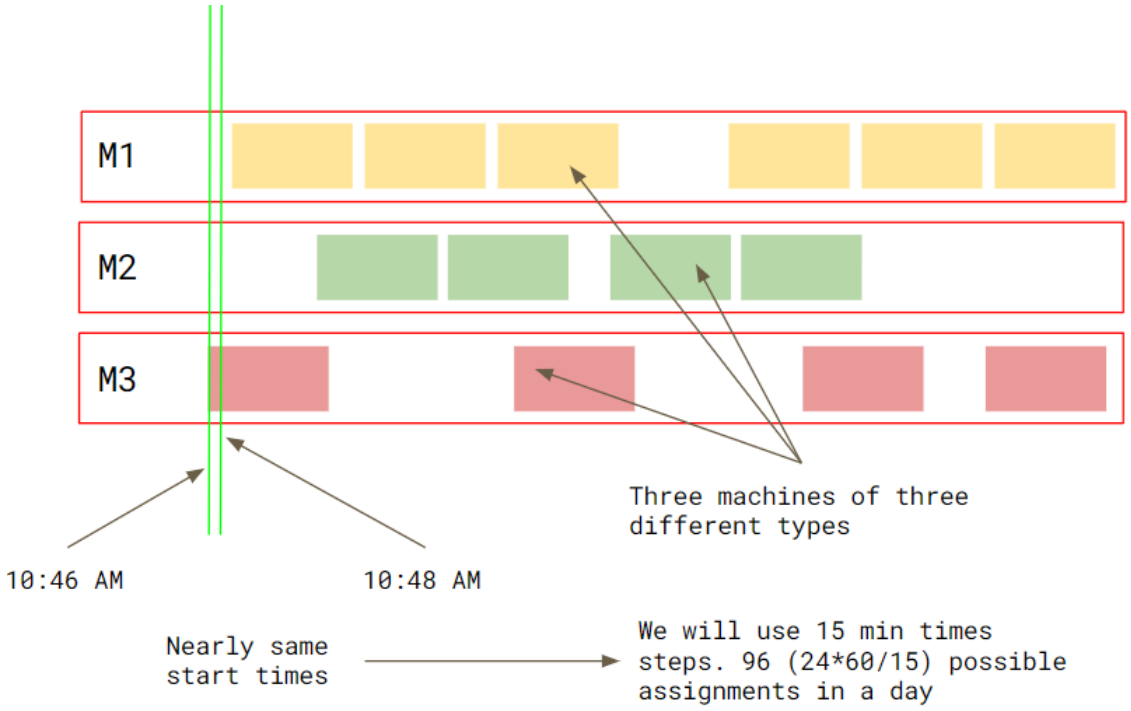


FIGURE 4.19: Genetic Space formulation

- We start with understanding the formulation of the genetic space with the above example. The red, green, and yellow colors represent one instance of each of the three distinct types of machines mentioned previously. We are representing the machine as a set of processes (group of jobs). Each process is further represented as a group of jobs. In the above diagram 4.19, each of the cells represents a single job that needs to be completed.

- We then assign time to each of the jobs, as can be seen from the figure 4.20. The entire day is first divided into α unit time steps (α is 15 minutes for course training and 5 minutes for fine-tuning in our experiments.) Each job is allocated the value of the time block that they fall in, where each block is α unit length.

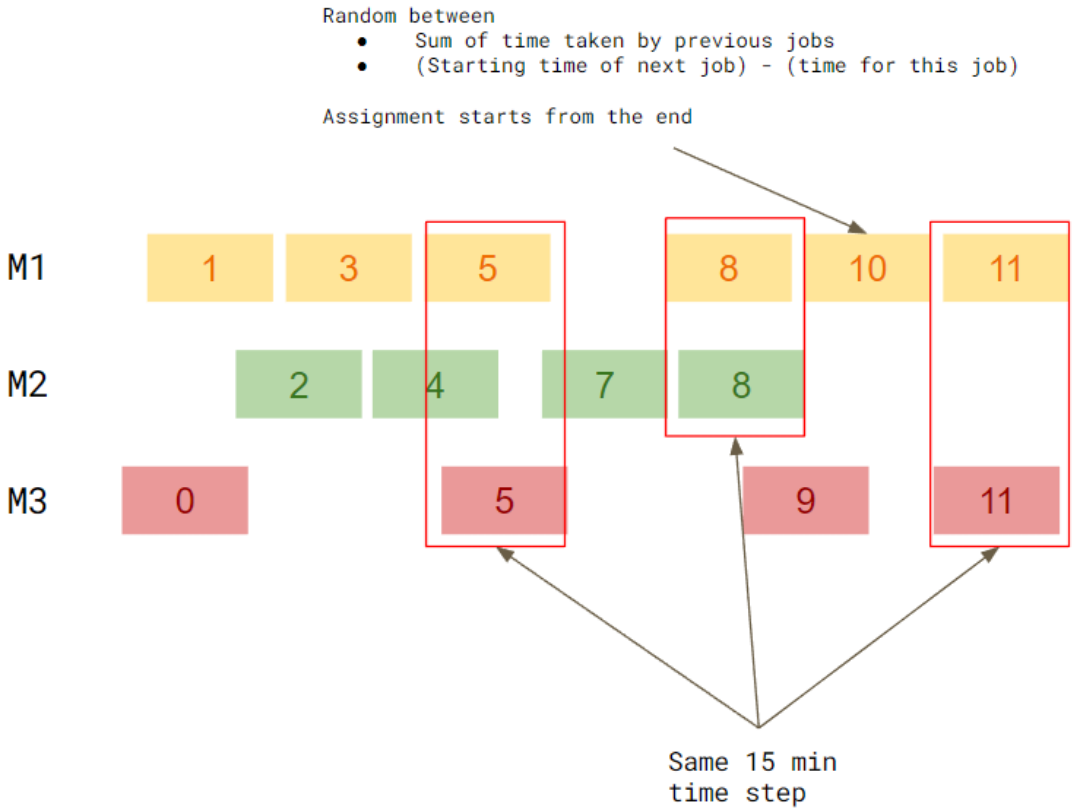


FIGURE 4.20: Adding time points to each schedule

- Once we have completed the assignment of the time to each process, we can now create the chromosome/gene from these sequences of time. We are already aware of which job goes to which machine from the metadata and the uniform assignment of each job.
  - Join the time sequence of each of the distinct types of machines by joining all the jobs on the machine together. This gives us a sequence of numbers representing the jobs that need to be completed in the order in which they need to be completed. This can be understood from figure 4.21
  - Now, we take the sequences formed in the previous step and join them together to form a single sequence. The order in which the sequences are joined does not matter significantly. We just need to make sure that each sub-sequence representing a single machine should not have any

reordering. This newly yielded sequence is our chromosome as can be seen in figure 4.22.

– Even though the order in which the sub-sequences are joined doesn't matter, we need to store the order of joining for reconstructing the schedule back from the chromosome.

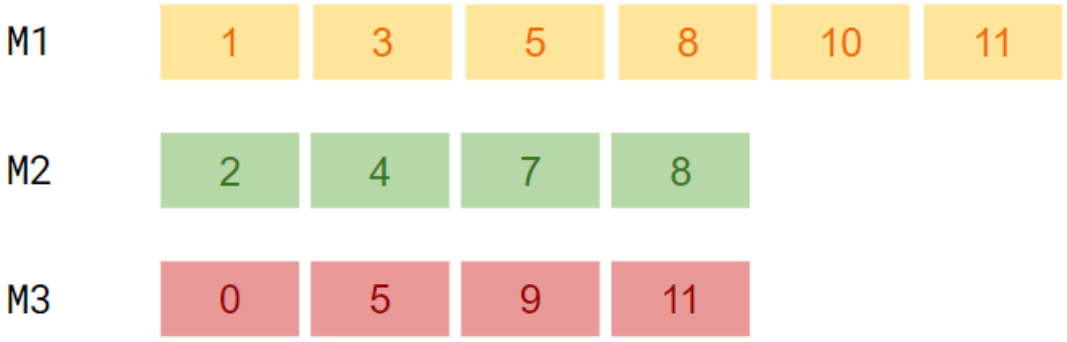


FIGURE 4.21: After joining the jobs to form sequence

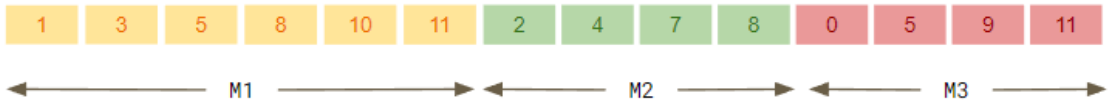


FIGURE 4.22: single gene representing the schedule

4.9.2 Mutation

Beginning with the first and most important process of the genetic algorithm for producing new offspring is **mutation**. For mutation, we will be performing the following mentioned steps to achieve a stable and valid mutation.

- Randomly select any one of the machines (of any type) on which the mutation needs to be applied.

- Starting from the ending of the sequence of the jobs to be completed on that machine (which is represented as a sequence of numbers corresponding to their scheduled slot), for each job on the machine, determine whether (probability = 0.5) or not to shift the job. Here we are considering only a backward shift at the moment, that means that we can only postpone the current scheduled job

- If we decide to shift the job in our current view, then we select a random number between

  – $current\_value + 1$, where $current\_value$ represents the current slot assigned to the job

  – (New start time of next job) - (Time taken for this job)

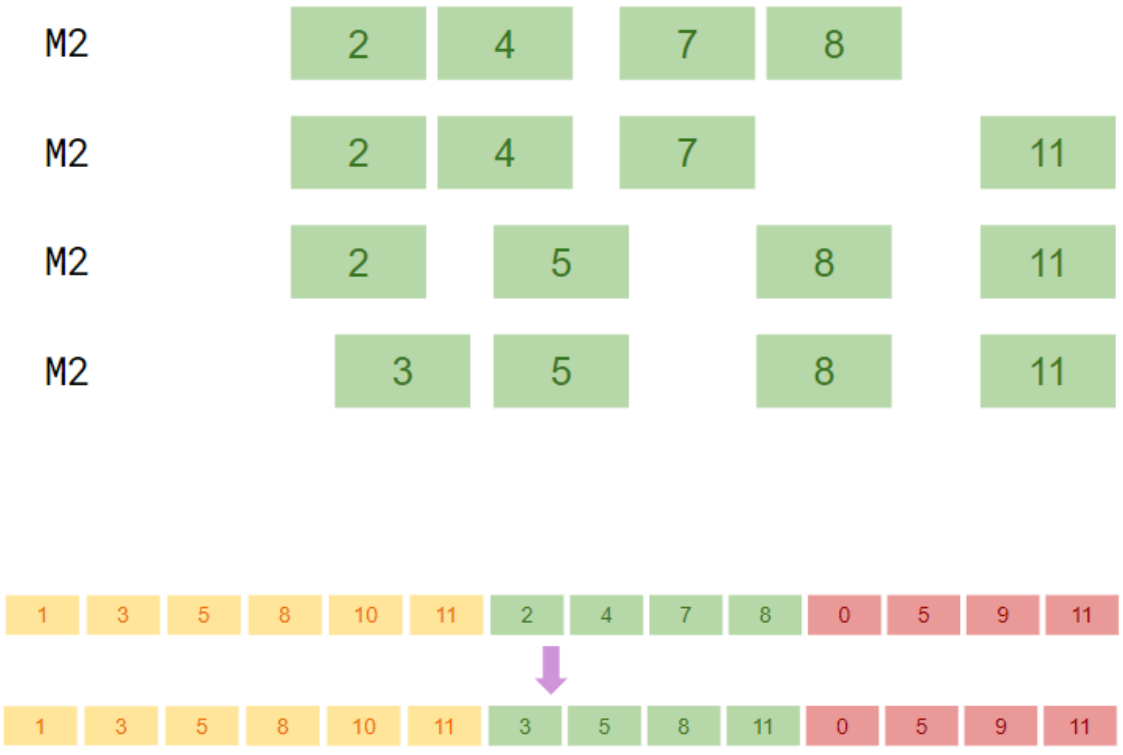


FIGURE 4.23: Mutation of a single gene

4.9.3 Crossover

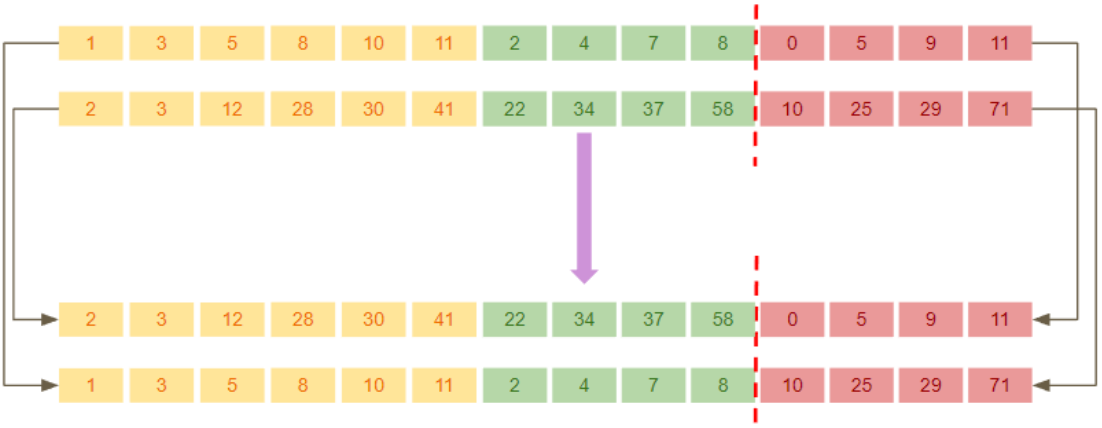


FIGURE 4.24: Crossover of 2 parents (Method-1)

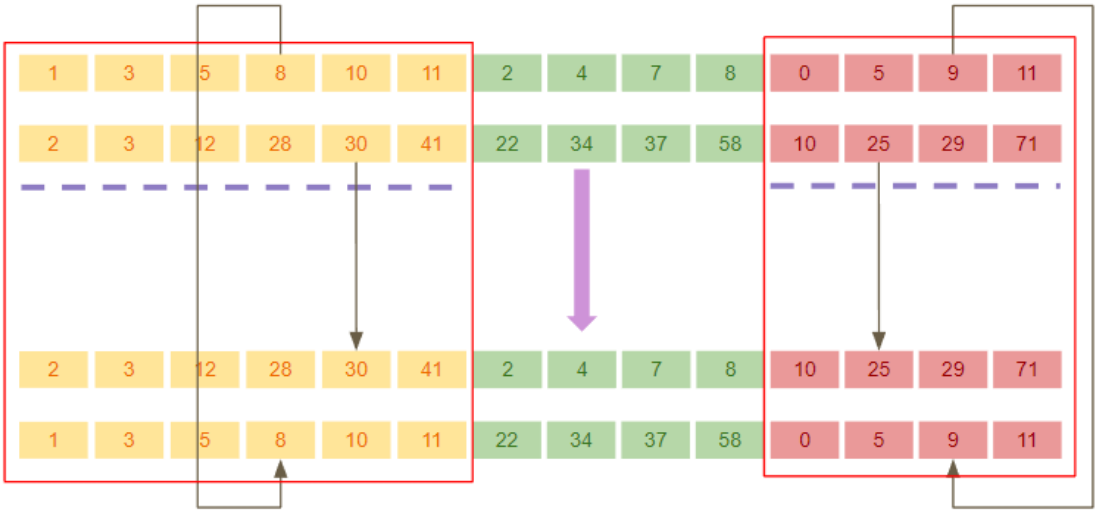


FIGURE 4.25: Crossover of 2 parents (Method-2)

Performing a gene crossover over two healthy parents is much more complicated as the generated schedule can yield either unstable offsprings, which means the schedule is not possible (for example, intersecting schedule or out of the limits of the day), or healthy parents me give rise to poor offsprings on performing crossover. We will primarily focus on two types of crossover, as mentioned below.

- **Crossover-1**: Figure 4.24 represents the first crossover scheme. Here we first select two healthy parents randomly. Then within these parents, we select a crossover point. The crossover point is selected such that it marks the joining point of two sub-sequences. If the point belongs to any of the subsequences then we may generate an unstable schedule. Hence it is quite necessary that the selected crossover point be the joining point of two sequences. Once the crossover point is selected, we iterate for all the jobs after the crossover point and swap the values of the respective cell of parent 1 and 2 (that is we perform $swap(parent[i_1][j], parent[i_2][j]))$. This process always yields a stable schedule after the crossover.

- **Crossover-2**: Figure 4.25 represents the second crossover scheme. Here we begin by selecting k segments out of the N segments which comprise the gene. Each segment here represents a schedule of a single machine. When we select a segment, then we take the entire segment into consideration. Then we iterate over each cell of the k segments that we just chose and for each of the cells of those segments, we do a swap on the values of the respective cell of the parents (that is we perform $swap(parent[i_1][j], parent[i_2][j]))$. This process is similar in essence to crossover-1 but is a more flexible scheme and is guaranteed to yield a stable sequence. Indeed after many iterations, the crossover-1 and crossover-2 schemes will perform equivalently.

4.9.4 Genetic Diversity

4.9.4.1 Genotypic Diversity

Genotypic diversity refers to the assessment of genetic variation within a population by examining the differences in the genetic makeup (chromosomes) of individuals. It is an important concept in evolutionary biology and genetics, as a higher level of genetic diversity can contribute to a population's ability to adapt to changing

environments, resist diseases, and maintain long-term viability. There are several methods to measure genotypic diversity, with two common approaches being the Hamming distance and Shannon entropy.

- **Hamming Distance:** This method involves calculating the average pairwise Hamming distance between the chromosomes of individuals in the population. The Hamming distance is a measure of the number of differing elements (e.g., nucleotides, alleles) at corresponding positions between two chromosomes. By computing the average pairwise Hamming distance for the entire population, one can obtain an estimate of the overall genetic diversity. A higher average distance indicates a greater level of diversity, suggesting that the population contains a wider range of genetic information.
- **Shannon Entropy:** Another approach to measuring genotypic diversity is to compute the Shannon entropy for each gene position (locus) across the entire population. Shannon entropy is a measure of uncertainty or disorder in a set of values, and in this context, it quantifies the variability of alleles at each gene position. Higher entropy values indicate a greater degree of genetic diversity at a specific locus. By summing up the entropy values for all gene positions, one can obtain an overall indication of the population's genotypic diversity. This method provides a more nuanced understanding of genetic variation, as it accounts for the distribution of different alleles at each locus rather than just the pairwise differences between individuals.

4.9.4.2 Phenotypic Diversity

Phenotypic diversity refers to the assessment of variation within a population based on the differences in the observable characteristics, or phenotypes, of individuals. These characteristics are the result of the interaction between an individual's genetic makeup (genotype) and environmental factors. Analyzing phenotypic diversity can

help researchers understand the potential adaptive capacity of a population, identify selective pressures, and inform management strategies in fields such as agriculture, ecology, and conservation. There are several methods to measure phenotypic diversity, with two common approaches being the Euclidean distance and variance or standard deviation.

- **Euclidean Distance:** This method involves calculating the average pairwise Euclidean distance between the phenotypic values of individuals in the population. The Euclidean distance is a measure of the straight-line distance between two points in a multidimensional space, where the dimension size is the same as the length of the chromosome and each dimension represents the physical trait of the job to be completed. By computing the average pairwise Euclidean distance for the entire population, one can obtain an estimate of the overall phenotypic diversity. A larger average distance suggests a higher level of diversity, indicating that the population contains a wider range of observable characteristics.
- **Variance or Standard Deviation:** Another approach to measuring phenotypic diversity is to determine the variance or standard deviation of the objective function values (e.g., fitness, productivity, or other relevant metrics) in the population. Variance is a measure of dispersion, quantifying the degree to which individual phenotypic values deviate from the mean value of the population. Standard deviation is the square root of the variance and is expressed in the same units as the phenotypic values. Higher values of variance or standard deviation indicate greater diversity, as they suggest a broader range of phenotypic outcomes within the population.

4.9.4.3 Species Count

Species count is a method for assessing diversity within a population by dividing it into subgroups or species based on specific criteria, such as similarity in genetic

or phenotypic characteristics. A higher number of species implies greater diversity within the population, which can be important for understanding the adaptive capacity and resilience of the population to changing environmental conditions or other selective pressures. One approach to performing species count is to apply a clustering algorithm, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which can automatically identify groups of similar individuals within the population.

DBSCAN is a density-based clustering algorithm that groups data points based on their proximity to one another and their density. It is particularly effective for identifying clusters of varying shapes and sizes, as well as detecting noise in the data. The key parameters for DBSCAN are:

- **EPS (Distance Bound):** This parameter defines the maximum distance between two data points for them to be considered as part of the same cluster. A lower EPS value results in smaller, denser clusters, while a higher value leads to larger, more dispersed clusters.
- **MinPts (Minimum Points):** This parameter specifies the minimum number of data points required to form a dense region. Data points within a dense region are considered part of a cluster, while those outside are treated as noise.

In the context of species count, the DBSCAN algorithm can be used to identify subgroups within the population by considering the Euclidean distance as the metric of the distance between individuals. This distance metric is particularly suitable for continuous data, such as phenotypic traits or genetic markers. To obtain optimal results, the EPS parameter should be fine-tuned to ensure that the identified clusters accurately represent the underlying structure of the population.

In the Python programming language, the DBSCAN algorithm can be implemented using the **sklearn.cluster.DBSCAN** module from the popular **scikit-learn** library. This implementation provides a convenient and efficient way to perform

species count and analyze diversity within a population. By applying the species count method and analyzing the resulting clusters, we can gain valuable insights into the population structure, identify distinct subgroups, and inform decision-making in areas such as conservation, breeding programs, and population management.

4.9.4.4 Fitness Diversity

Fitness diversity evaluates the variation within a population based on individuals' fitness values. It helps us understand the potential for adaptation and informs decision-making in areas like genetic algorithms, breeding programs, and population management. Two common approaches to measuring fitness diversity are:

- **Average Fitness:** Calculating the average fitness value of the population and monitoring its changes over time can provide insights into the overall fitness progression and selective pressures. Typically, a diverse population would have individuals with varying fitness levels, and observing the average fitness can help researchers identify stagnation in the evolutionary process.
- **Fitness Range:** Determining the range of fitness values (difference between maximum and minimum fitness values) can reveal the level of diversity and presence of highly fit or low fit individuals within the population. The fitness range is calculated as the difference between the maximum and minimum fitness values in the population. A larger range indicates a higher level of diversity, as it suggests that the population contains individuals with a wide spectrum of fitness levels.

Chapter 5

Results And Discussions

5.1 Heuristic Algorithms

5.1.1 Single Agent Environment

Following Algorithms are used to set base lines for the upper and lower bound of the model's performance

- **First Come First Served**: the First Come First Served algorithm replicates the human behaviour. That is as soon as we see a free machine and we want have the job to be completed we place the job on the machine with complete disregard to the cost at that moment of the day.
- **Lower Bound (based on Binary Search)**:
  - This algorithm provides an optimized lower bound to the bill amount that can be achieved.
  - This is ideal because the algorithm disregards the fact that there may be delay on behalf of the human user to schedule the job.
  - In this algorithm we will will perform a binary search on the limit to bill amount for per unit time and try to minimize it.

Symbols

- N = Number of machines
- J = Number of jobs
- T = Total time units in a day (units in our case will be 5 minute interval)

Algorithm 1 First Come First Served

Require: $N \geq 0, J \geq 0, T \geq 0$
  $cur \leftarrow 0$
  $bill \leftarrow 0$
  while $cur \neq T$ **do**
    $cur \leftarrow cur + 1$
    while Some Job Finished Now **do**
      $N \leftarrow N + 1$
    end while
    power ← power consumed by machines working currently
    bill ← bill + cost(power, cur)
    while $N > 0$ and $J > 0$ **do** ▷ Job remains and machine is free
      $N \leftarrow N - 1$ ▷ Greedy scheduling machine
      $J \leftarrow J - 1$ ▷ Greedy scheduling jobs
      Update Finish time of the Job
    end while
  end while

Algorithm 2 Binary Search based Heuristic (Lower Bound)

Require: $N \geq 0, J \geq 0, T \geq 0$
  $L \leftarrow 0, H \leftarrow 0$
  while $L \leq H$ **do**
    $M \leftarrow (L + H)/2$
    if CheckScheduling(M) **then** ▷ If returns 0 then scheduling is not possible
      $H \leftarrow M$ ▷ limit is sufficient
    else
      $L \leftarrow M + 1$ ▷ limit is insufficient
    end if
  end while
  bill = CheckScheduling(L)

Algorithm 3 Check Scheduling is Possible

  $cur \leftarrow 0, bill \leftarrow 0$
  while $cur \leq T$ **do**
    $power \leftarrow 0$
    while $cost(power, cur) \leq M$ **do**
      if $N > 0, J > 0$ **then**
        $N \leftarrow N - 1, J \leftarrow J - 1$ ▷ Shift the previous scheduled block if required
        Update Power, Bill ▷ use **cost()**
      end if
    end while
  end while
  Return `bill` if all the jobs were schedule else 0

5.1.2 Multi-Agent Environment

The multi agent experiment are performed with the following configurations

Configuration for multi-agent (machine and jobs category wise)						
House	Machine 1	Machine 2	Machine 3	Job 1	Job 2	Job 3
House 1	6	5	1	10	16	4
House 2	2	4	1	3	11	1
House 3	10	2	5	4	5	12

5.2 Greedy (First Come First Served)

- The algorithm, in this case, remains the same as the single agent scenario, with the additional modifications that now different houses will also be scheduling together.
- The First come, First Served approach here highlights the condition that the neighbors are unaware of what other houses in the building are consuming. So whenever there is a job to be done, and a device is available, a job is scheduled on the device.
- Due to the greedy scheduling, adding deadlines will have no effect, only re-ordering in the order of the job allocation based on deadline-based priority. The following image visualizes the schedule of the job during the entire day.

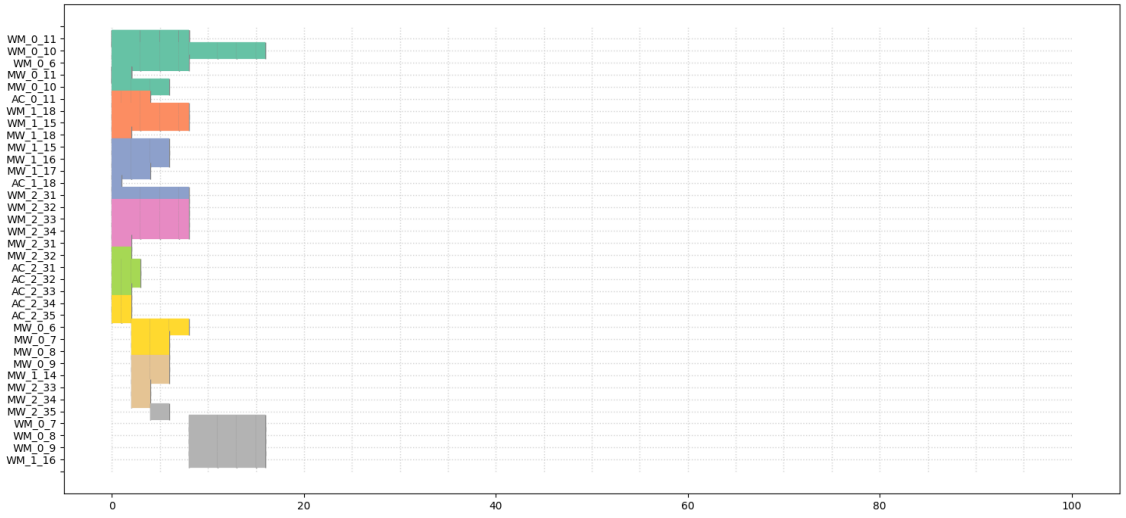


FIGURE 5.1: Scheduling of the jobs during the day by greedy algorithm (FCFS)

5.3 Binary Search combined with Knapsack Solver

- As the scale of the environment grows into the multi-agent setting, simple reordering of the jobs within the given power/cost bound becomes extremely difficult. Hence we use a new approach in this setting. Within the given bound on the cost on which the binary search is being performed, we use all the jobs that can be scheduled at the current time step and pass it to an efficient knapsack solver (from Google OR-Tools) to figure out which jobs should be scheduled at the current time step to get the optimal value.

- The knapsack solver assigns a value "one" to each job in the scenario in which there is no deadline, primarily because the completion of the job is important here. As shown in figure 5.2, there is a uniform trend in which the jobs have been assigned to different machines, primarily because peak power is uniformly distributed over the entire day.

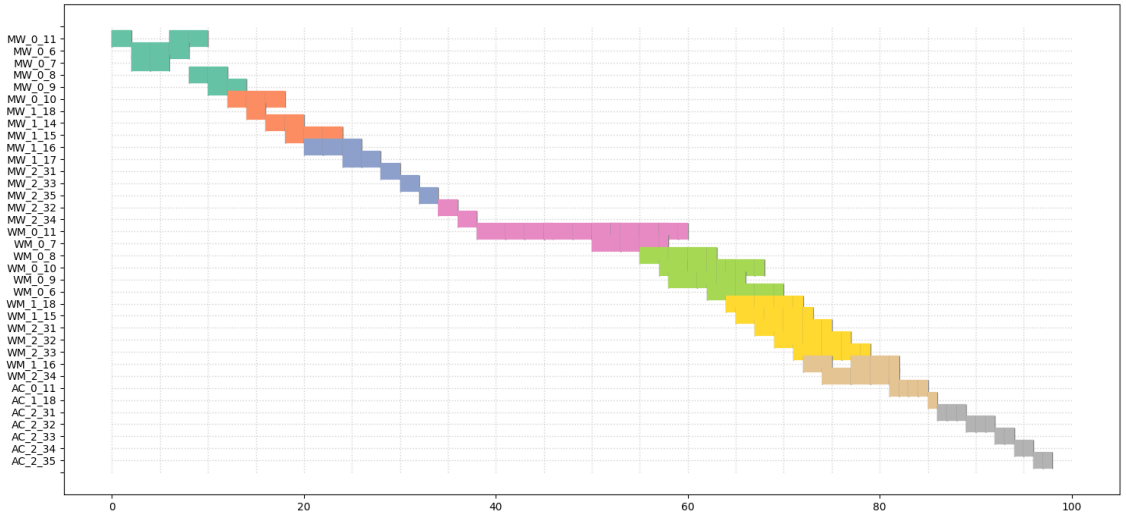


FIGURE 5.2: Scheduling of the jobs during the day by Binary Search combined with Knapsack solver (without deadlines)

- Including deadlines is a challenge in this proposition of the algorithm, primarily because within the bound decided by the binary search, we are allocating all the jobs that are waiting to be scheduled, and the knapsack solver while picking the jobs doesn't give priority to them based on deadlines. Hence we devise a new idea to include in the deadline. We change the values of each job that

the knapsack solver assigns based on how near we are to the scheduled job deadline. Primarily, the value of a job can be viewed by the following equation (LIMIT=INF):

$$Minimum(LIMIT, 1/(deadline - time_{now})^4)$$

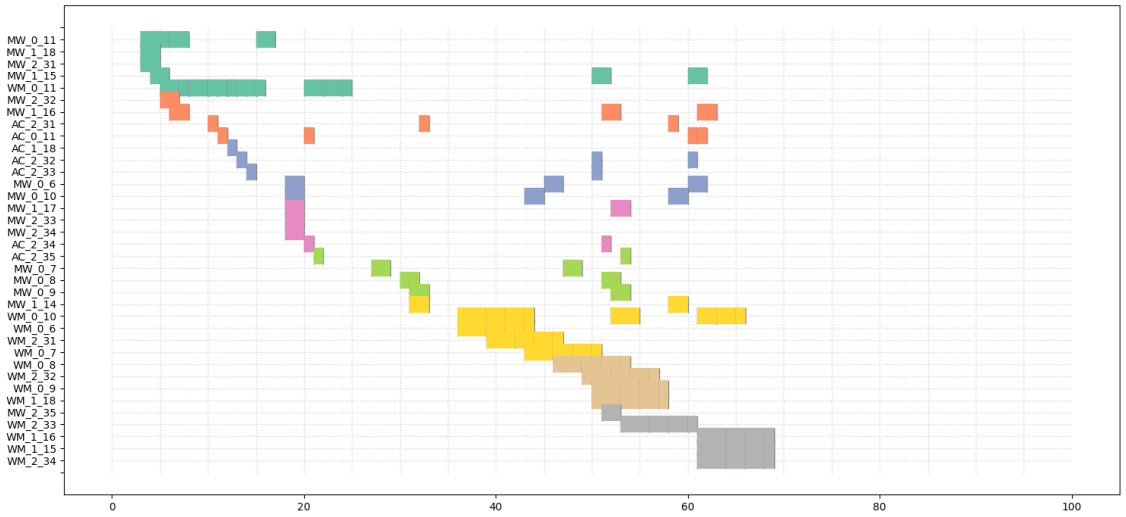


FIGURE 5.3: Scheduling of the jobs during the day by Binary Search combined with Knapsack solver (with deadlines)

5.4 Mixed Integer Programming

- We draw inspiration from the disjunctive mixed integer programming formulation described in Ku and Beck 2016. The decision and fixed variables can be defined as follows:

  1. x_{ij}, which is the integer start time of job j on machine i
  2. d_j, is the duration of the j^{th} job.
  3. D_j is the deadline of the j^{th} job.

- Constrain formulation consists of the following

  1. $0 \leq x_{ij} \leq T$
  2. $x_{ij_1} + d_{j_1} \leq x_{ij_2}, \forall j_1, j_2 \epsilon M_i$

- Without the presence of deadlines, we see that MIPS stacks all the jobs in that part of the day when the cost for each unit is the lowest. But this leads to a significant spike in the usage of electricity during a very short duration of the day. If a similar trend is followed by a lot of buildings in the town, the generators will soon start overheating and malfunctioning. In due course, the price during that time of the day will be increased.

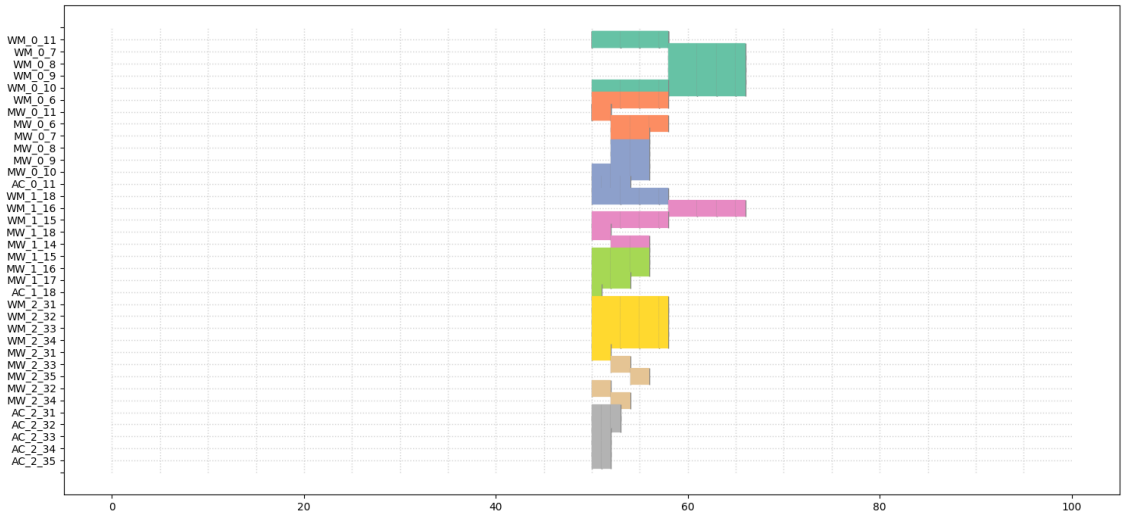


FIGURE 5.4: Scheduling of the jobs during the day by MIP(without deadlines)

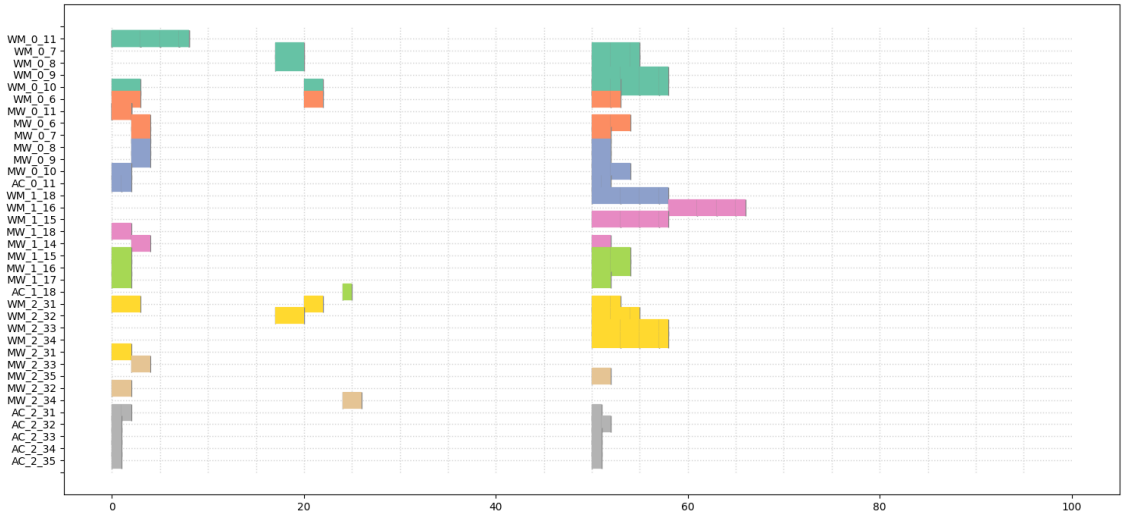


FIGURE 5.5: Scheduling of the jobs during the day by MIP(with deadlines)

- Adding deadline to the Mixed Integer Programming based solver was a much easier task as compared to the binary search solver. The only change required is to set the following constraint to the decision variable.

  1. $0 \leq x_{ij} \leq D_j$

- From figure 5.5 we see that now, the mixed integer programming solver tries to stack each job to the lowest possible cost region within the deadline. this will still lead to spike at different point in time of the day, but given the constraints, the solvers finds the minimum bill amount that can be achieve, and this actually gives a optimal lower bound to compare with the multi agent reinforcement learning approaches.

- This is a lower bound primarily because it does not take into account the uncertainties of the human behaviour, that there might be delays during scheduling. Adding uncertainties while train reinforcement learning model so that agent can model near real world scenario, will always add some disturbance to the lower bound.

5.5 Training Details

Two methods of training are implemented and their results are compared in this report. In each of these methods we focus on two training instance, one after 20,000 steps (69 episodes), and other after 2 million steps (6940 episodes).

5.5.1 Full Data Training

- The model results that are provided in the subsequent slides were trained on 15 machines each and 40 jobs to be performed by each class of the machines.

- We train the RL Agent on higher number of machines, and the end user will be limited to add up to 15 devices from each class. On request from end user the number of devices may be increased.

- If we are adding lesser number of machines then we just need to set the cells corresponding to assignment to invalid cells to `-inf` (Black colour cells in the diagram).
- Some auxiliary functions, like those for prioritized experience replay, are being used from the stable baseline (fork of the OpenAI baselines https://stable-baselines3.readthedocs.io/en/master/ repository), since they have quite extensively optimised implementations.

5.5.2 Incremental Training

- This incremental training method starts with a single hidden layer in the feed-forward network and trains it for one machine.
- As we keep increasing machines in the following steps, we add a new layer for each new device.
- When we add the second machine, we will add a new hidden layer between the output layer and the final fully connected layer, which we have already trained for one device.
- The training after adding a new machine will majorly affect only the new layer added to the fully connected network and minorly to other layers (due to the freezing of layers). Hence, training will be faster.
  - Before starting the training during the initial warm up steps, the model was also run a few times on some of the initial experience accumulated with some random states, so that the feature extractor layer does not start training with completely random kernel initialization. This has proven to improve the performance.
  - During training the model allowed previous layers to be trainable to 10 - 20 % of the steps. That is, if I say I am training for the 3rd layer for 3000 steps, then I allow 2500 steps for training the third layer and 500 steps for training the first two layers.

– Number of steps allocation was proportional to $\lceil layer_{number}^{1.5} \rceil$ (eg 1:3:6). Finally for fine tuning the model was run for 2000-5000 steps with the last two layers and the output layer set as trainable layers.

5.5.3 Predictor Network

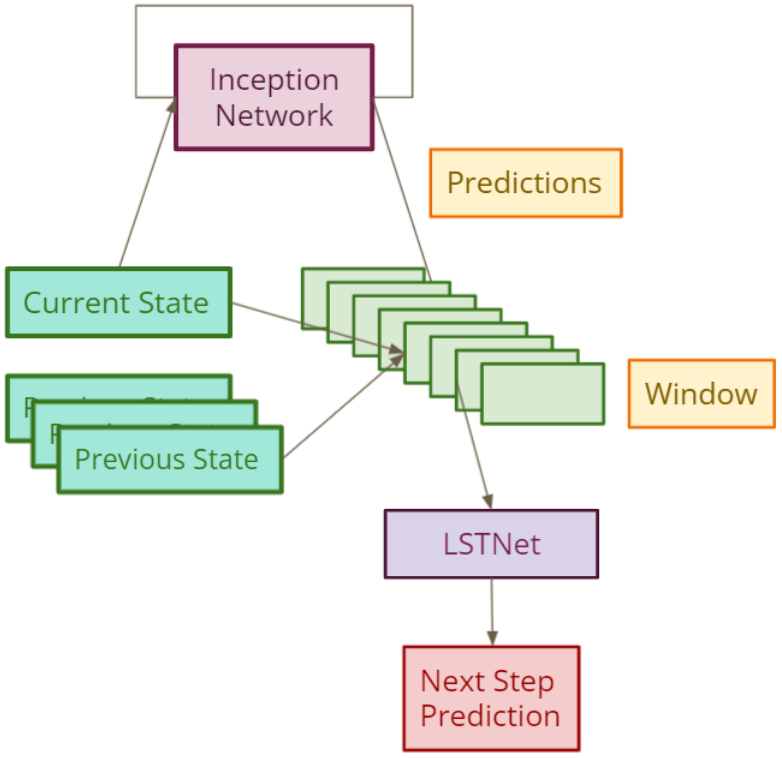


FIGURE 5.6: Predictor Network used to improve training results

We make use of a predictor network which is a trained inception network (since we observe in the results, that the inception network had provided best performance as compared to the other feature extractor networks). The LSTNet Lai et al. 2017 requires a windowed data as an input, so as to capture the features in the time series. The autoregression link in the LSTNet network 4.14 assumes that the future time predictions is only dependent on the previous time steps. But in our case we exploit the learned inception network to exploit few steps in the future, so that the model gets an idea of a possible good future on taking an action. We use window of size 9 time steps, out of which 6 are historical (previous) time step and 3 possible

future time step. Historical time step data is available from the buffer containing the stored prior experiences while training the DQN.For the future time steps we use the inception network based agent to predict next three states and give the output after simulation. The main reason being that the inception network is already trained for giving output with good results.

5.6 Features Added to Model

- **Delayed Scheduling**: While training the environment has been configured so that, with probability 0.5 the environment will delay the scheduling decision taken by the machine between 1 and 5 minutes. This is to take into account the delay in scheduling by the end user.
- **Dynamically Adding Jobs**: A function (add_job) has been implemented so that additional jobs can be added dynamically by the end user at any time of the day. This does not need any retraining of the agent because we have already trained the agent for more number of jobs and the end user can add any number of jobs below the limit of maximum jobs that can be added.
- **Dynamically Adding Machines**: A function (add_machine) has been implemented so that additional machines (belonging to already added distinct types of machines) can be added dynamically by the end user at any time of the day. This also does not need any retraining of the agent because we have already trained the agent for more number of jobs and the end user can add any number of machines up to the limit.

5.7 Dataset Available

- Synthetic Data has been synthesised from original dataset belonging to different events using the process to generate synthetic dataset for household devices described in Klemenjak et al. 2020.
  - Electricity usage by Devices and the energy load by the devices
  - Articulate synthetic jobs to be scheduled on the Devices from the load profile of the device.
- **Dataset Available**
  1. **Pecan Street Data**: Dataport hosts all the data collected via Pecan Street's (Texas USA) water and electricity research.

2. **Individual Household Electric Power Consumption Dataset:** measurements gathered in a house located in Sceaux (7 km of Paris, France) between December 2006 and November 2010 (47 months).
3. **Household Power Consumption:** Individual household electric power consumption dataset collected via sub-meters placed in 3 distinct areas of a home in France

5.8 Results and Observations (Single Agent RL))

- From the figure 5.7 we can see that the Episode reward increase with the number of steps taken. This trend is a standard Reinforcement Learning Agent Learning trend, and thus we are assured that the model is learning.
- From the figure 5.8(a) we realise that the loss of the model is decreasing with time, that means that the neural network is fitting for our data.

Note: The colour coding in the following figure represents the different runs of training the agent to observe that is the model working correctly for .

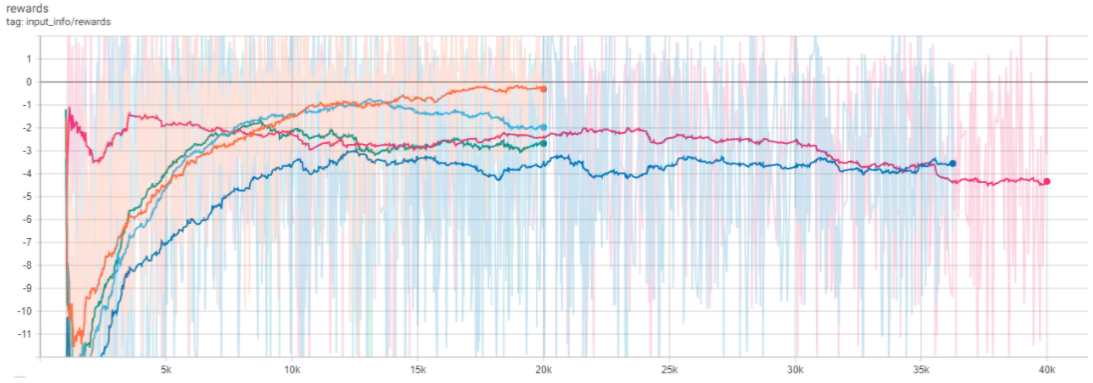
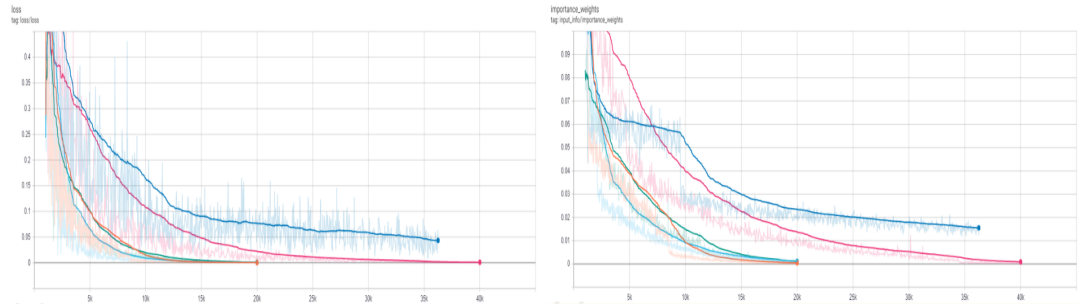


FIGURE 5.7: Episode Rewards



(a) Training Losses

(b) Importance Weights

FIGURE 5.8: Training Statistics

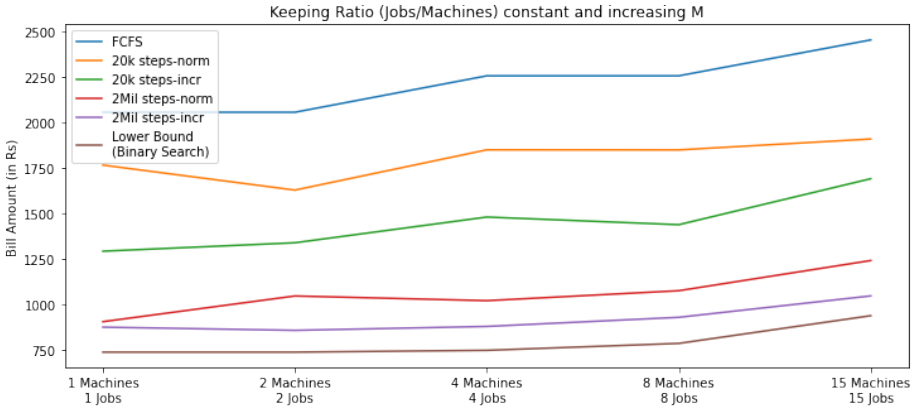


FIGURE 5.9: Bill Amount using the same number of machines and jobs

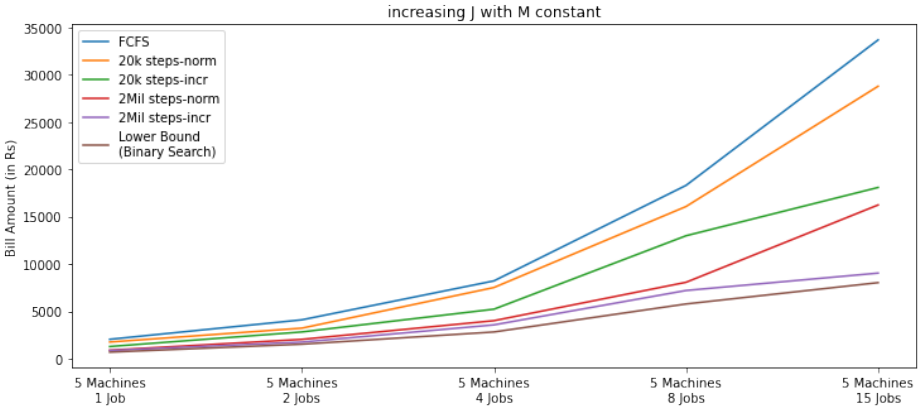


FIGURE 5.10: Incrasing Jobs keeping Machines constant

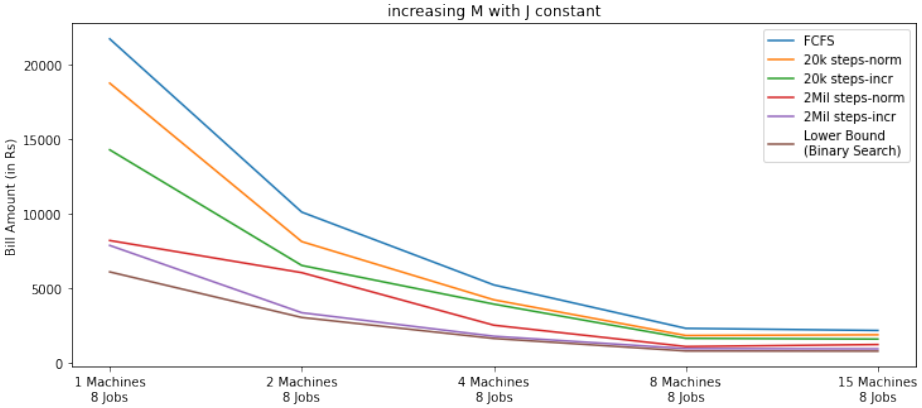


FIGURE 5.11: Incrasing Machines keeping Jobs constant

Bill Amount for figure 5.9 (Rs) ($\frac{\#Devices}{\#Jobs} \doteq 1$)					
Algorithm or Steps	1 Devices 1 Jobs	2 Devices 2 Jobs	4 Devices 4 Jobs	8 Devices 8 Jobs	15 Devices 15 Jobs
FCFS	2056.51	2056.51	2256.51	2256.51	2453.51
20k steps-norm	1766.58	1629.45	1850.4	1849.92	1909.86
20k steps-incr	1294.15	1340.46	1481.59	1439.62	1691.85
2Mil steps-norm	907.46	1048.47	1022.74	1077.36	1242.99
2Mil steps-incr	877.38	859.83	881.34	931.33	1048.9
Binary search	740.25	740.25	750.25	788.25	940.2

- From figure 5.9 we see that the trend of bill amount almost remains constant with increasing the jobs (or machines) keeping the ratio of J/M constant. However there is a slight increase in Bill amount with increasing the number of machines (or jobs). this is due to the fact that when many machines start working simultaneously especially when ToD pricing of electricity is low, there is higher risk of crossing the penalty limit, thus the bill increases.

Bill Amount for figure 5.10 (Rs) (increasing Devices)					
Algorithm or Steps	1 Devices 1 Jobs	2 Devices 2 Jobs	4 Devices 4 Jobs	8 Devices 8 Jobs	15 Devices 15 Jobs
FCFS	2026.21	4113.02	8226.04	18309.66	33671.8
20k steps-norm	1748.28	3233.16	7525.71	16077.94	28780.97
20k steps-incr	1277.35	2832.72	5237.26	12991.38	18091.85
2Mil steps-norm	908.88	2048.47	4022.74	8077.36	16242.99
2Mil steps-incr	887.23	1759.83	3581.34	7211.33	9048.9
Binary search	701.25	1540.85	2830.97	5788.25	8040.25

- From figure 5.10 we see that the Bill amount increases with increasing the number of jobs. This is mainly due to the reason that more the number of jobs more time the machines will rune, hence more power is consumed, so the bill increases. Similar reasoning can be given to 5.11. Increasing the number of machines provide better opportunity to the RL agent for properly scheduling the jobs on different machines to minimize the cost. Thus the output scheduling is in accordance with the hypothesis.

Bill Amount for figure 5.11 (Rs) (increasing Jobs)

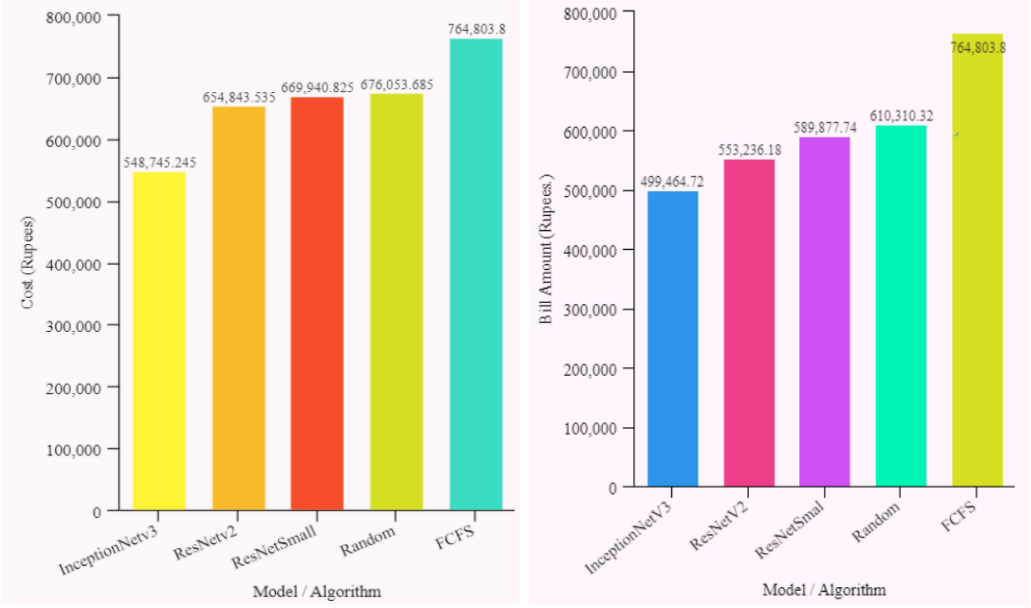
Algorithm or Steps	1 Devices 1 Jobs	2 Devices 2 Jobs	4 Devices 4 Jobs	8 Devices 8 Jobs	15 Devices 15 Jobs
FCFS	21737.13	10113.02	5226.04	2309.66	2167.8
20k steps-norm	18766.58	8133.16	4225.71	1829.95	1878.7
20k steps-incr	14294.15	6532.72	3937.26	1640.12	1591.35
2Mil steps-norm	8207.46	6048.47	2522.74	1090.26	1224.34
2Mil steps-incr	7877.38	3359.83	1781.34	950.53	934.89
Binary search	6100.25	3040.85	1630.97	790.99	780.22

- Most important observations from the above figures is that the incrementally trained models provide better result than the full data model when trained for the same number of steps. Also the model trained for more number of steps (2 million) provides better results. The incremental model trained for 2 million steps even performs comparable with our optimal heuristic too.

5.8.1 Results for Feature Extractor Network

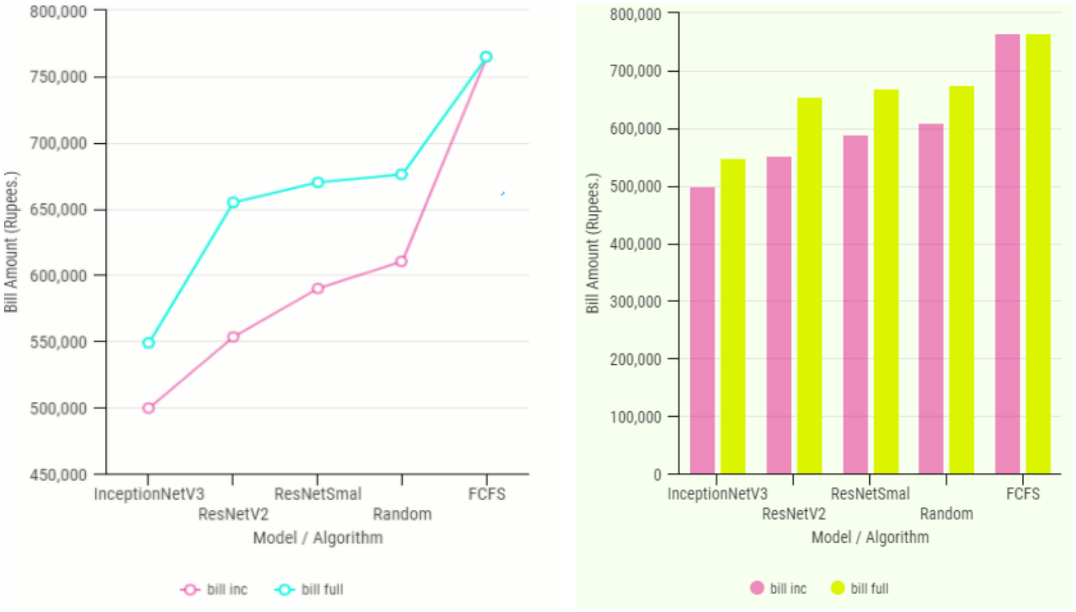
For the testing of the feature extractor network we schedule 15 jobs to be completed on 5 machines. We make use of the Inception Net V3 4.11, ResNet50 4.12 and ResNet Small 4.13 for feature extractor testing and compare the policy learned using these network against our FCFS and Random Policies. All of these models are trained for

2 million steps (5 min time steps) which is also equivalent to 60K steps (for 15 min time steps) for both incremental and full training.



(a) Full Training (b) Incremental Training

FIGURE 5.12: Electricity billing for one day's job



(a) incremental vs full training bill amount (line graph) (b) incremental vs full training bill amount (bar chart)

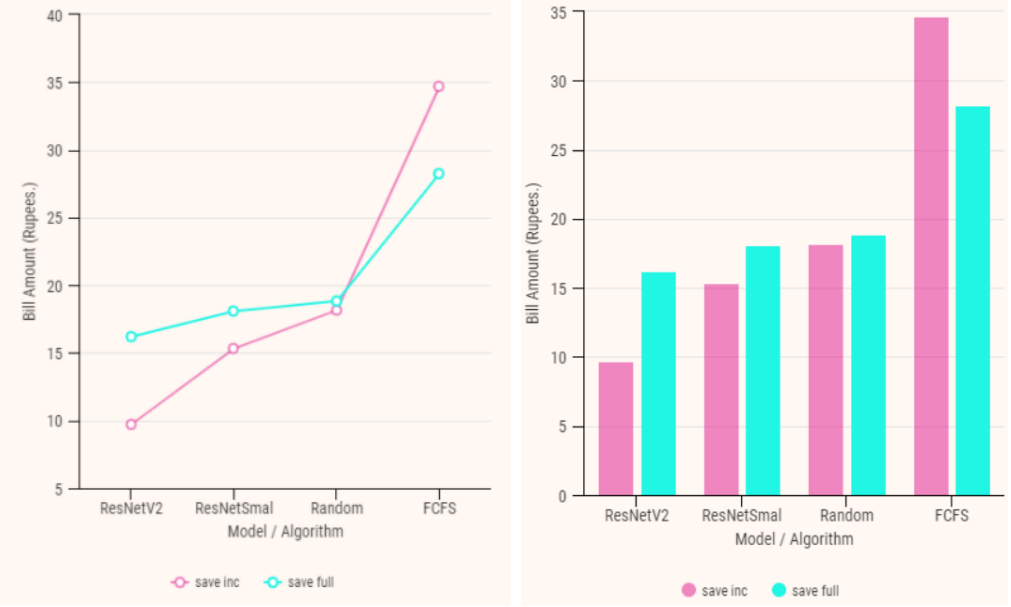
FIGURE 5.13: Comparison of the Electricity billing

The above graph 5.12 show the billing incurred by the user in one day using the schedule provided by the RL agent using different feature extractor networks. It is clear from both the plots 5.12(a) and 5.12(b) that the inception net has out performed the other feature extractor network in recognizing the patterns and accurately prediction the action to be taken in the near future for achieving best possible scheduling. Hypothesis for these results is that InceptionNet is compute optimized network and our model needs to map out a computational structure from the image, hence the InceptionNet has better efficiency in easily extracting the mathematical features from the image.

FIGURE 5.14: Comparision of time taken for a single step during training

Model check-pointing is used to store the intermediate models and restore models for starting next phase in the incremental training. Check-pointing and restoring the model from the checkpoint of huge models take more time hence incremental training can take quite huge time. In the above image 5.14 we give a comparative study of the time taken for the full training and incremental training. We use fixed number of checkpoints (every 2000 steps) in this case for both training paradigms. This graph 5.14 displays the time taken for 1 step in the training process by the different policies

used in DQN Agent. Compute optimized inception network is about 5 times faster than the ResetV2(50) model. Since Each step takes lesser time, the entire training time is overall faster for Inception Net as compared to other models.



(a) Bill amount saved (line graph)

(b) Bill amount saved (bar chart)

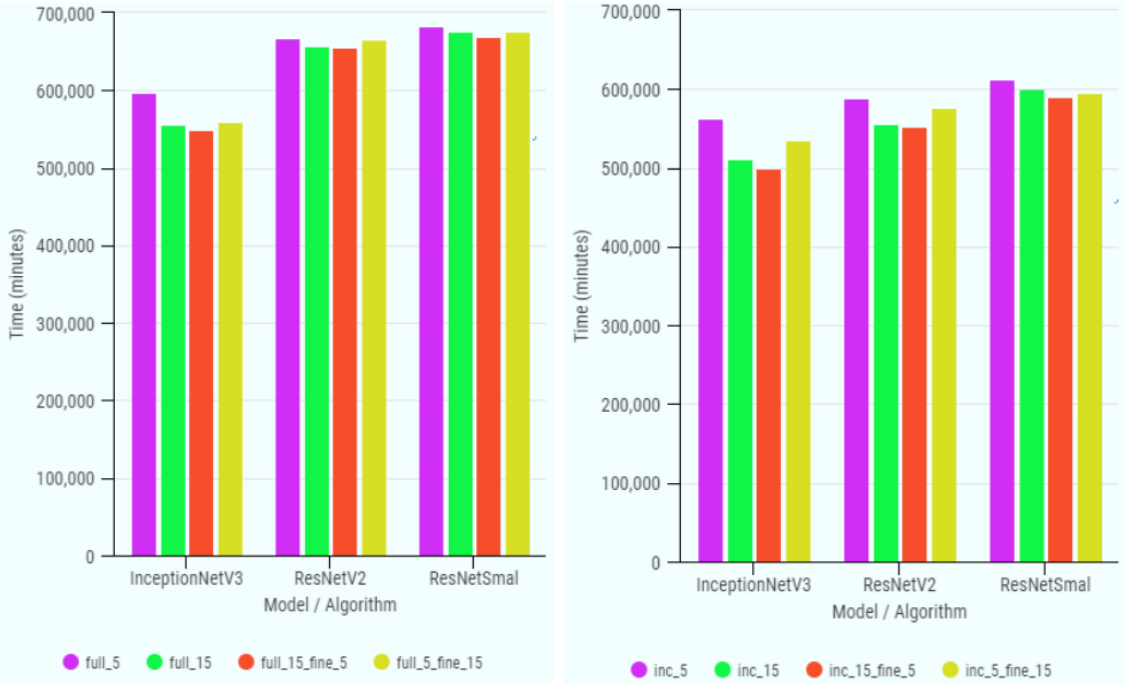
FIGURE 5.15: Comparison of the Bill amount saved by Inception Net as compared to other feature extractors

This graphs 5.15(a) and 5.15(b) displays the percentage of the bill amount that the inception net has saved as compared to the other algorithms / models. using incremental and full training. It is clear that inception net out performs the FCFS algorithm, which replicates the human behaviour by a large margin, and also has significantly better performance as compared to the other models.

5.8.2 Course Steps and Fine Tuning

In this section we enumerate the results of training the model using 15 minute time steps, 5 minute steps, 15 minute steps and fine tuning using 5 minute steps, 5minute time steps and then fine tuning on 15 minute steps. Here we divide the day into broad time steps of 5 or 15 minutes instead of 1 second mainly to reduce the observation space over which the scheduling will occur. Training with 15 minute

steps and fine tuning on 5 minute steps mainly signifies that we first make our RL agent learn to schedule the devices if we had broken the day in 15 minute time steps (i.e only multiple of 15 minutes can be allocated to a process). Once it has learned a scheduling on 15 minute time steps a day, then we tell the model that now it has to learn scheduling on 5 minute time steps a day.



(a) Full Training (b) Incremental Training

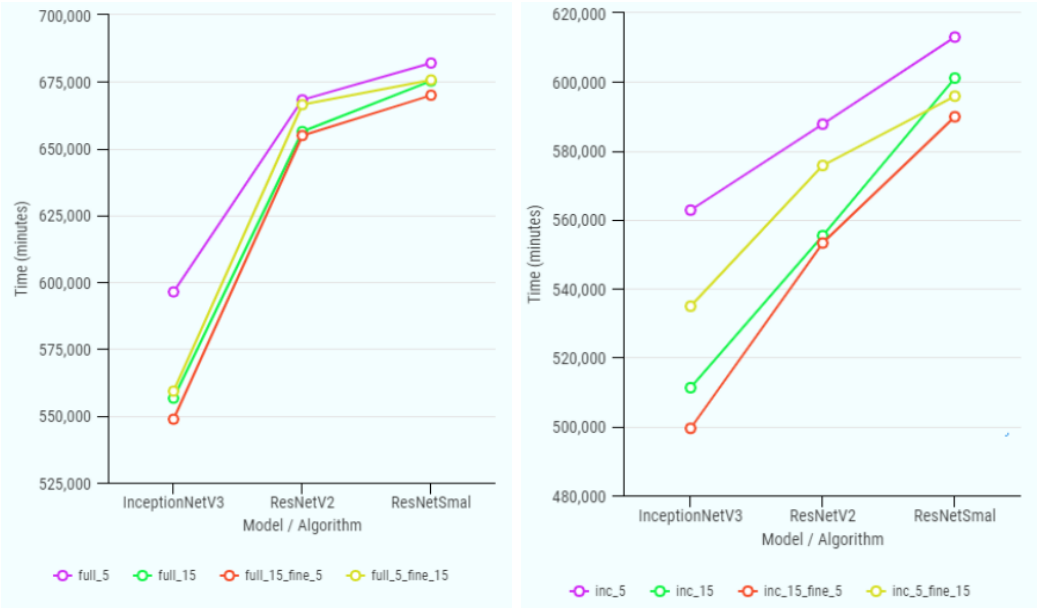
FIGURE 5.16: Fine tuning results with different time steps

This graph 5.16 shows the bill amount achieved by different policies on the following four types of unit steps with incremental training

- Day is divided into steps of 5 minutes and then trained
- Day is divided into steps of 15 minutes and then trained
- Trained first on steps of 15 minutes then fine tuned on 5 minutes steps.
- Trained first on steps of 15 minutes then fine tuned on 5 minutes steps.

From the plots 5.17 we can see that for the case of full training the difference between using fine tuning and not using fine tuning is not that significant, however when we

go to incremental training we can see significant effect on the cost saving with the use of incremental training. We also observe that training with 15 minute course steps and fine tuning with 5 minute steps has better performance than doing viz-a-viz. We also observe that in general with any form of tuning, the incremental training has better performance than full training.



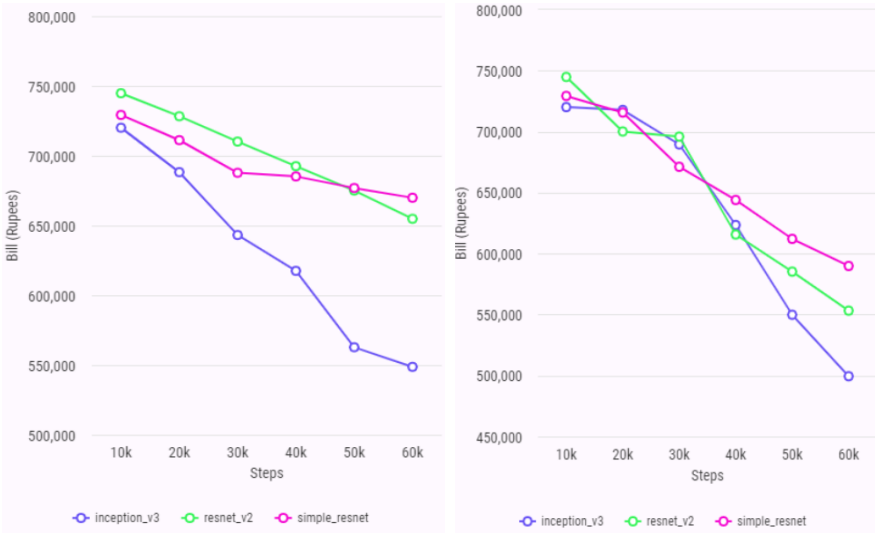
(a) Full Training (b) Incremental Training

FIGURE 5.17: Comparison of fine tuning model with different time steps

5.8.3 Happiness results

- From the graph 4.17 it is evident that we have made the happiness function has been made continuous, this helps model training as the model will not be seeing any sudden jump in the rewards.
- FCFS scheduling achieved a happiness score of -316.11 with bill amount of Rs.7,64,803.
- However incrementally trained Inception Net to maximize the happiness achieved a score of 389.23 with a bill amount of Rs.5,13,116
- This is greater than incremental training without maximizing happiness, but better than discontinuous happiness) (All deadlines near the 12 noon or the mid day)

5.8.4 Step wise comparison



(a) Full Training (b) Incremental Training

FIGURE 5.18: Comparison of results of full and incremental training step-wise

The graph 5.18(a) and 5.18(b) shows the performance of the model using full and incremental training for 60K steps, respectively. The model evaluation is done after every 10K steps, and the computed minimum bill amount (after running 5 times) is stored as the evaluation result for performance measure. For full training, we observe that the inception 4.11 network consistently performs better than the resnet models. The ResNet50 4.12 performs worse than ResNet small4.13 initially but learns the feature eventually and performs better than a smaller network. On the other hand, for the incremental training, we see that it initially performs almost equivalently since the lower steps represent training for a lesser number of machines for all the networks and hence lesser observation space. But as the observation space becomes bigger with the addition of more machines, the inception network performs better, as seen in the earlier results.

5.8.5 Recurrent Network and Predictor results

From figure 5.19(a), we see that in the case of inception, network addition of an LSTM layer did not lead to much improvement in performance, mainly because it had already learned the feature more efficiently. However, there is a significant improvement in the performance of the ResNet50 model. From figure 5.19(b), we see that LSTNet lone, when trained, doesn't perform much better. Indeed it performs worse than inception net, though it has outperformed residual networks. However,

on adding the predictor network (which is a trained inception net v3), we see that there is a significant improvement in the performance, and the newly learned policy even outperforms the policy learned with Inception Net alone. We see that augmenting some good possible future possibilities (peek in the future) with the past experience gained (peek in the past) has better performance. However, the predictor network's performance also plays a crucial role in giving good predictions to make better decisions for future time steps. From figure 5.20 we also observe that adding a predictor network to train the models reduces the peak power consumed and smooths the household's power consumption, which reduces the bill.

(a) BiLSTM results (b) LSTNet Result

FIGURE 5.19: Comparison of results of adding Recurrent connection

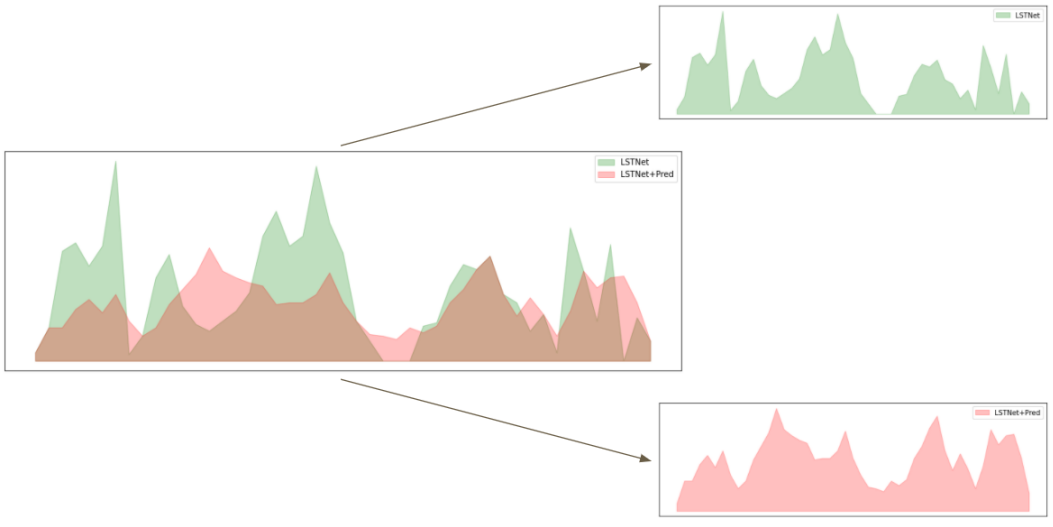


FIGURE 5.20: Improvement in Peak power consumption with LSTNet

5.9 Results and Observations (Multi-Agent RL)

5.9.1 Ablation study of heuristics

In all the following results of the ablation study, we have considered cost bound. But the cost bound and the power bound are directly proportional and are related by the pricing function determined by the electricity company providing the services. Clearly, if the power usage goes up, the cost goes up and viz-a-viz. Hence the cost and power bound have been used interchangeably.

- For all possible bounds, the results of the FCFS (greedy strategy) always remain the same, primarily due to the deterministic nature of the algorithm. The cost incurred by FCFS is **39532.5**. This provides an upper bound on the efficiency of the RL model since the models should not perform worse than this. But the FCFS strategy achieves 100% job completion within the deadline.

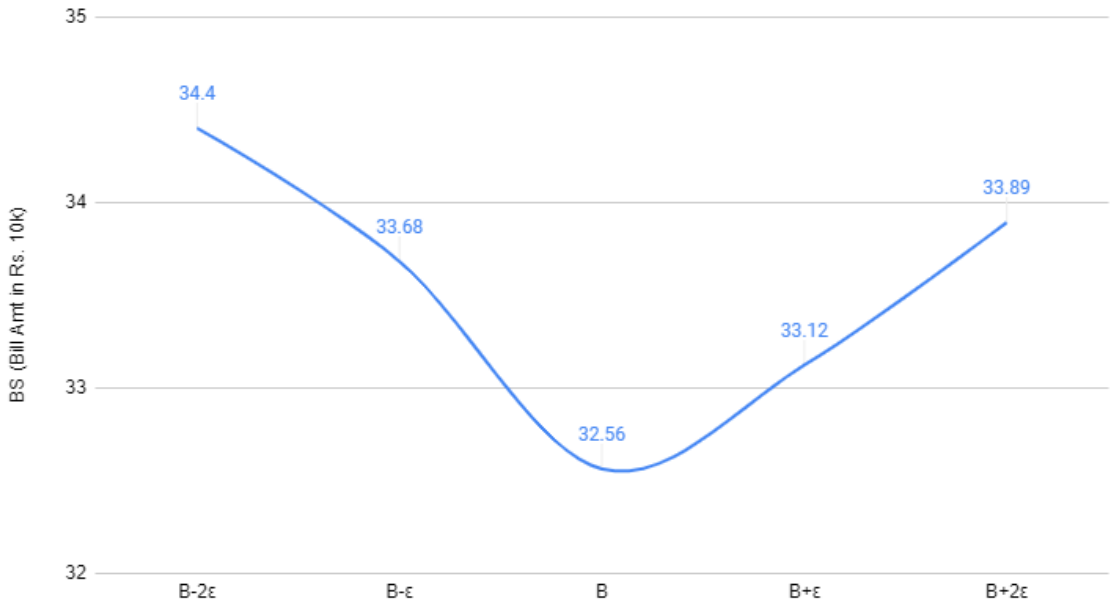


FIGURE 5.21: Bill amount incurred versus the different cost bound. B = 623 (optimal bound)

- The binary search algorithm achieves a cutoff bound of **Rs 623** per time step for the answer, which gives optimal bill amount while distributing the power bound equally. We now try to disturb the bound by multiple of ($\epsilon = Rs.10$),

simultaneously giving a little relaxation on scheduling for exceeding the bound at some time step; we see that we have an overshoot in the bill amount achieved on both directions. The increase in bill amount as the power bound decreases satisfy our intuition, but there should be s drop in the bill amount when the power bound is increased. But we see an increase in the bill amount because of the greedy nature of the way we are allocating the jobs to the knapsack. If, at some time, the knapsack sees that within the bound, there is some space to schedule the job it will schedule it and not look into the future for whether there is another schedule that can reduce a job's cost of energy consumption.

- The Mixed Integer Programming solver always tries to stack the jobs in the best possible location as per the lowest cost per unit of energy. According to intuition, the stacking will be better if the per time unit bound is more because more jobs can be accommodated in the slot with lower cost, and hence the total bill amount over the day comes down. The results achieved also validate or assumption, as seen in 5.22.

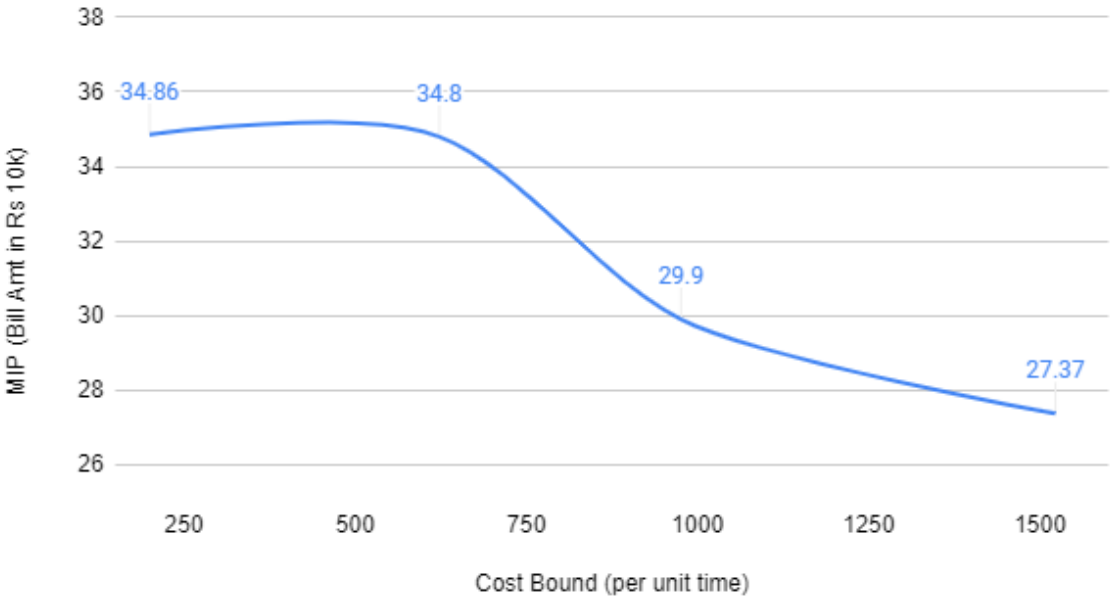


FIGURE 5.22: Bill amount incurred versus the different cost bound for MIP solver

- Another interesting observation in these heuristics can be observed in the setting that with a cost cutoff of Rs. 200 per time step, the Mixed integer programming solver is able to find a solution in which all the jobs were scheduled during the day but the Binary Search based solver is not able to do so. This

is also partly due to greedily allocating jobs to Knapsack solver and also due to the fact that not much rearranging of the jobs is taken into account with binary search.

5.9.2 Centralized Training for Decentralize Execution

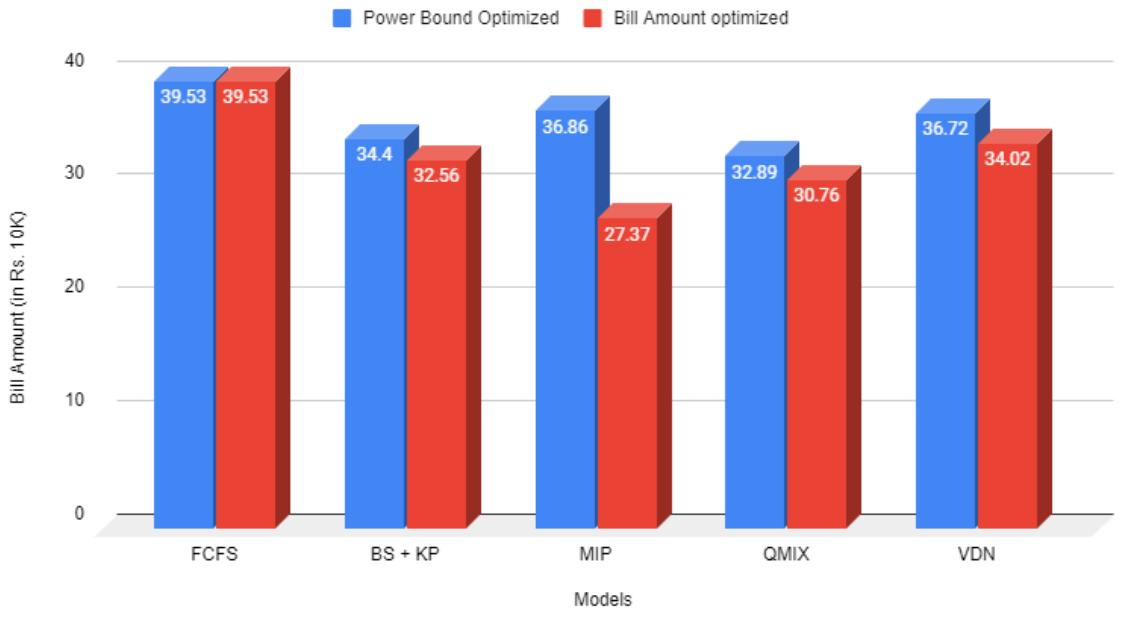


FIGURE 5.23: Bill amount incurred versus the different models (a) Peak power optimization setting (b) Bill Amount optimization setting

- In the results for the Reinforcement Learning models, the change between the Bill optimization and peak power optimization settings differs in the way the reward function are modelled in the environment simulation. Hence based on the reward maximization, the policy learnt is also different.
- Value-Decomposition Network model performs surely better than FCFS (Greedy) model, and is better than random policy since it adheres to the deadline and completes all the tasks assigned to it within the day before the mentioned deadlines.
- It is clear from the figure 5.23 that QMIX out performs other value algorithm. The monotonicity constraint on the action value function of the QMIX Rashid et al. 2018 may also reinforce the result as to why the binary search algorithm

and QMIX (when trained with pea power optimization settings) converge on results which are very close. Both algorithms enforce monotonicity constraints, just in different perspectives.

- The QMIX is actually able to reorder the time step wise limit/bound, which is being used by the binary search algorithm more efficiently by increased exploration and efficient reordering of the jobs during the day to reach to a much better solution.

- QMIX also models the uncertainty of the human scheduling error and hence the optimal bill amount achieved by the QMIX is a bit astray from the optimal bill amount achieved by the lower bount set using Mixed Integer Programming solver.

- The table below gives the values of the Bill amount and the Peak Power required (real time) during the schedule of the entire day, under different optimization settings, namely Bill Optimized (BO) and Peak Power Optimized (PO). From the table we see that, QMIX is not only able to reduce the bill amount, but also reduces the real time peak power, while striking a balance among the two.

Peak Power (W) and Bill (Rs.1K) for different optimization

MODEL	Bill[BO]	Peak[BO]	Bill[PO]	Peak[PO]
FCFS	39.53	89.66	39.53	89.66
Binary Search	32.56	65.29	34.40	56.50
Mixed Integer Prog.	27.37	76.41	36.86	43.13
VDN	34.02	73.83	36.72	70.75
QMIX	30.76	69.95	32.89	53.18

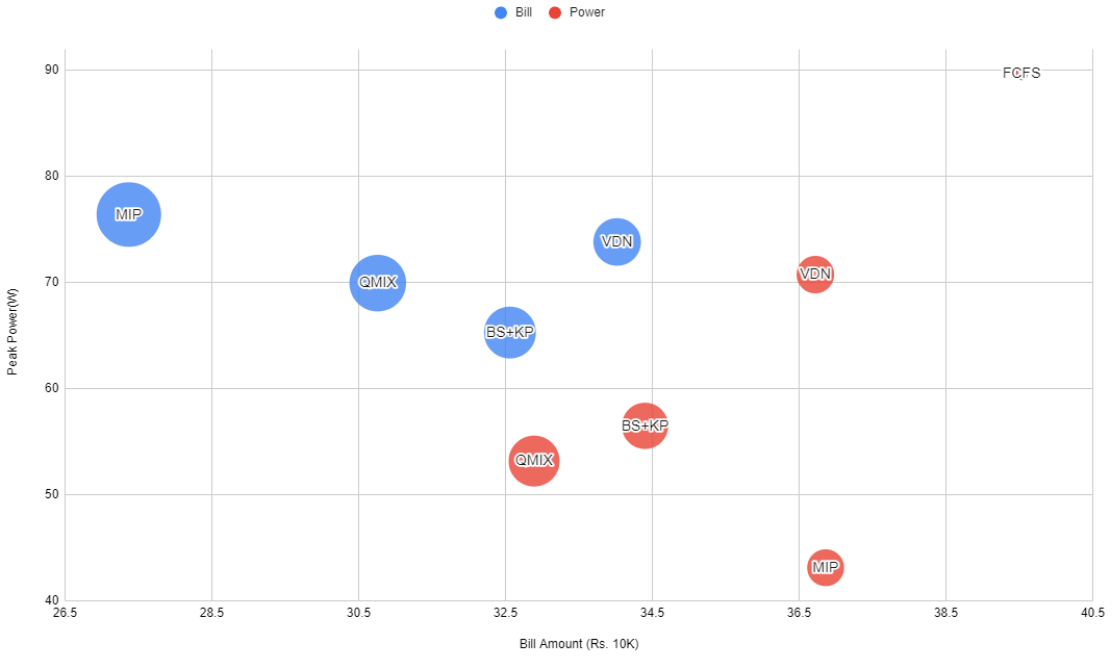


FIGURE 5.24: Scatter plot showing the position of the models on the Peak Power vs. Bill amount plane

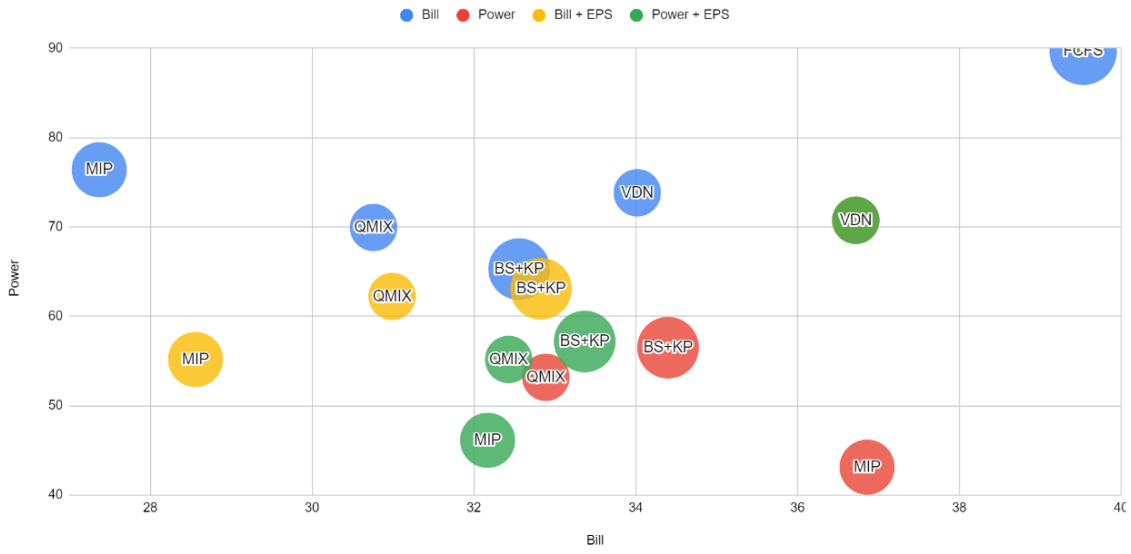


FIGURE 5.25: Scatter plot showing the position of the models on the Peak Power vs. Bill amount plane with increased sampling points

From Figure 5.25, we start figuring out that when we are only studying the effect of using different levels of optimization constraints, like high bill optimized, highly

power optimized, os skewed from bill optimized, and skewed from power optimized, a unique pattern starts to emerge. This pattern is enhanced when we outline the curve they follow as in Figure 5.26. These trend lines formed by the different algorithms seemingly form contours where each contour may demarcate some physical feature of the algorithm, like the mathematical limitation or resource constraint handling capacity. The trend lines are interestingly quite hyperbolic looking, and the increase in efficiency of one of the axis parameters clearly affects the performance of the other axis parameters, which also aligns with our ideas of the performance of the model. FCFS doesn't have contours as it is a completely deterministic algorithm. Hence all executions of the algorithm yield the same value in the absence of the introduction of any human-induced uncertainties.

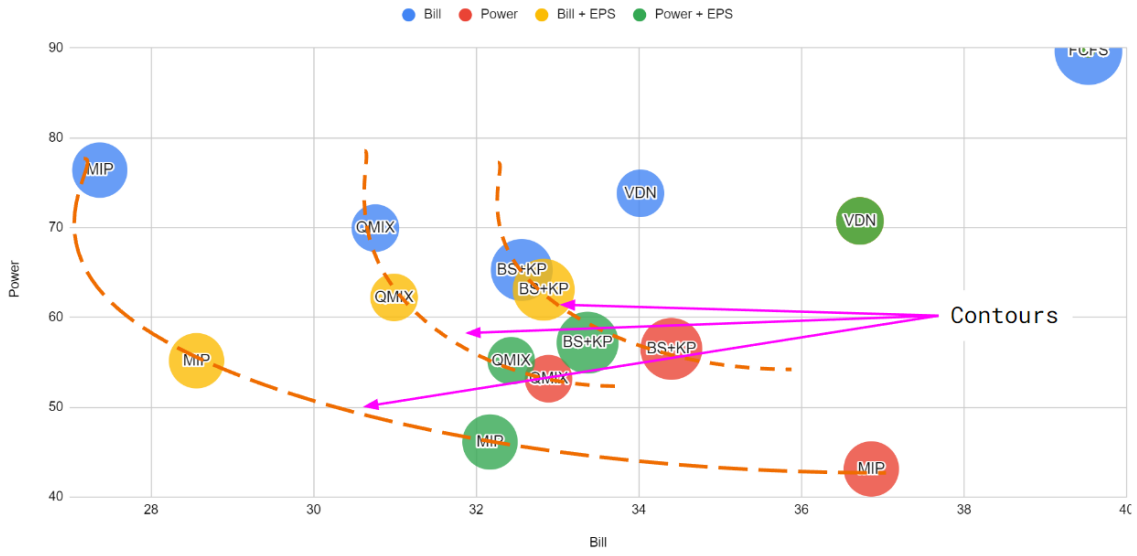


FIGURE 5.26: Outlining trends in 5.25 (Bill in Rs.1K, Peak Power in kW)

Following this, we test extensively by introducing the Happiness Function, as shown in Figure .5.27. From Figure 5.27, we see that for the deterministic counterpart algorithms like the binary search and mixed integer programming, the introduction and optimization of happiness didn't affect the results obtained. We see a slight increase in the bill amount and the peak power mainly because to optimize the happiness of the end-users, we need to relax the bill amount and peak power constraints to come extent. In the case of the reinforcement learning algorithms, we see a significant effect (compared to the more deterministic counterparts) on the increased bill amount

and the peak power. This supports that the models are trying to provide the best satisfaction to the users by optimizing the satisfaction scores.

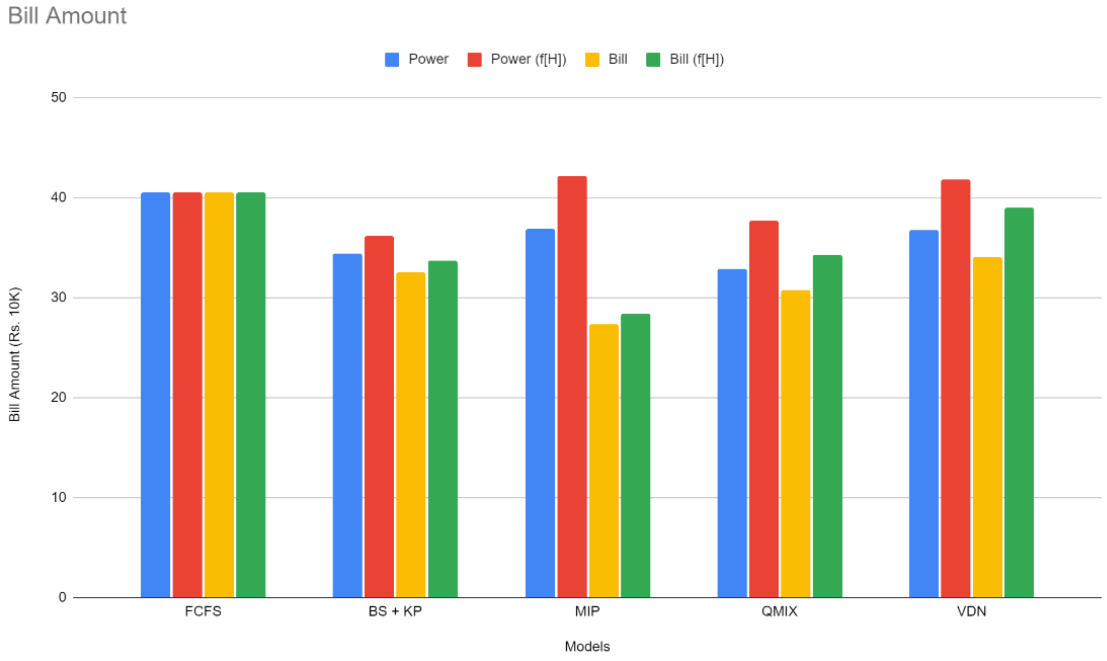


FIGURE 5.27: Comparison of models after introducing Happiness (f[H])

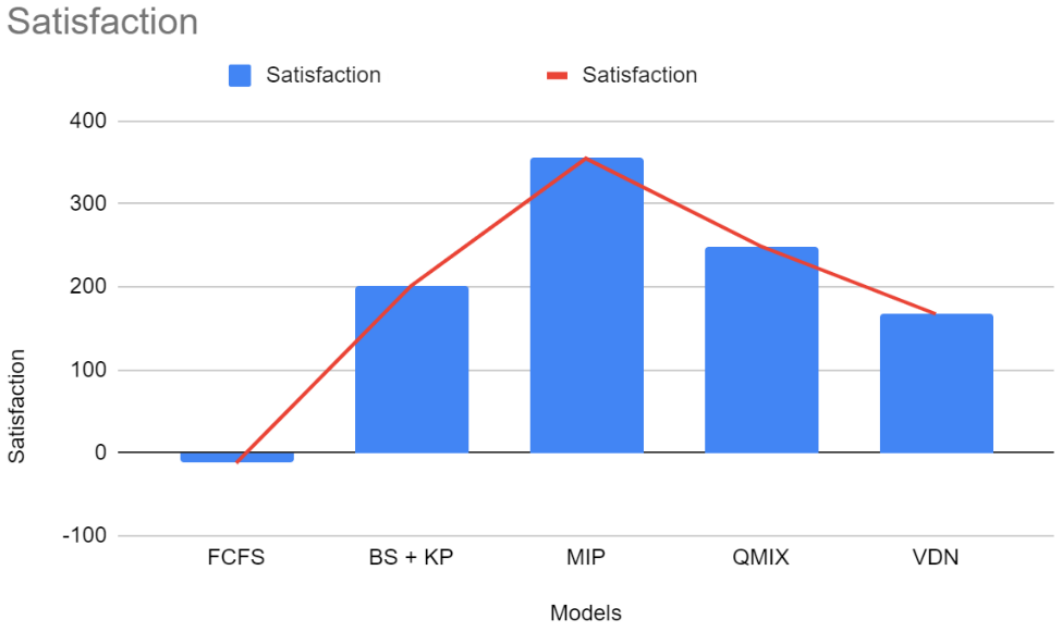


FIGURE 5.28: Satisfaction Scores achieved by different models

Figure 5.28 shows the satisfaction scores obtained by different models. As expected, the satisfaction score of the human nature based greedy "first come, first served" based algorithm will not be the most optimal, primarily in the cases where the job requirement near the deadline is more satisfactory, like food heating. The MIP model has a peak satisfaction score amongst all the models resulting from the performance of its efficient mixed integer programming algorithms. However, our reinforcement learning algorithm performs much better than the deterministic counterparts like QMIX, outperforming Binary Search in satisfying the end user.

5.10 Sensitivity Analysis

5.10.1 Revisiting the pricing schemes

We will now understand the pricing schemes we used to find the average results and the motivation behind selecting the pricing schemes. Different pricing schemes in this section have taken inspiration from different sources and have different significance. It is important to use different types of pricing schemes so that we can analyze and study the effects of our scheduling on both generic and specific cases.

5.10.1.1 Non-Monotonic Pricing Schemes

These are the category of the non-monotonic pricing schemes, which means they do not show any fixed trend, and their plot may or may not seem to fluctuate. This type of plot usually focuses on the different requirements at different times of the day and tries to control the usage during the peak times when most users are statistically available.

Figure 5.29 shows the first pricing model used in most of the above-mentioned results. The article of K Raheja Group 2018 inspires this pricing model. This pricing scheme is inspired by the studies that support that electricity use is particularly high during the morning to afternoon part, which is the peak office duration and the post-evening time when most people are at home. We want to control the peak usage in the household during the morning hours to shift in the early morning hours and or

late afternoons and similarly for the late evening hours. Hence, the pricing model has steeply high pricing during the morning to early afternoon and the late evening compared to the early morning and late afternoon to early evening duration.

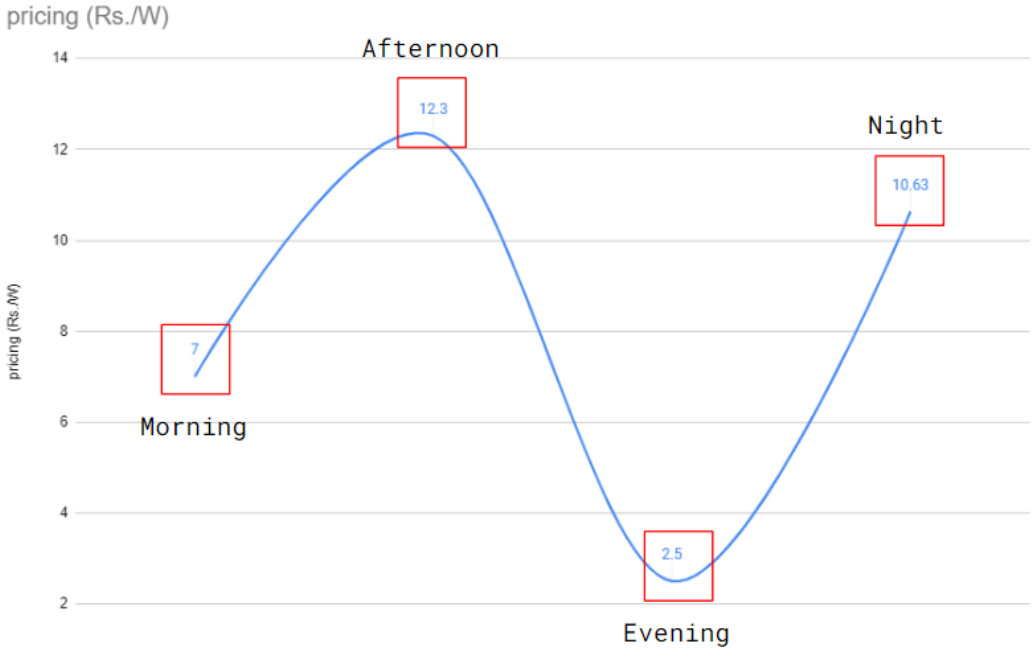


FIGURE 5.29: Pricing Scheme 1 [K Raheja Group 2018]

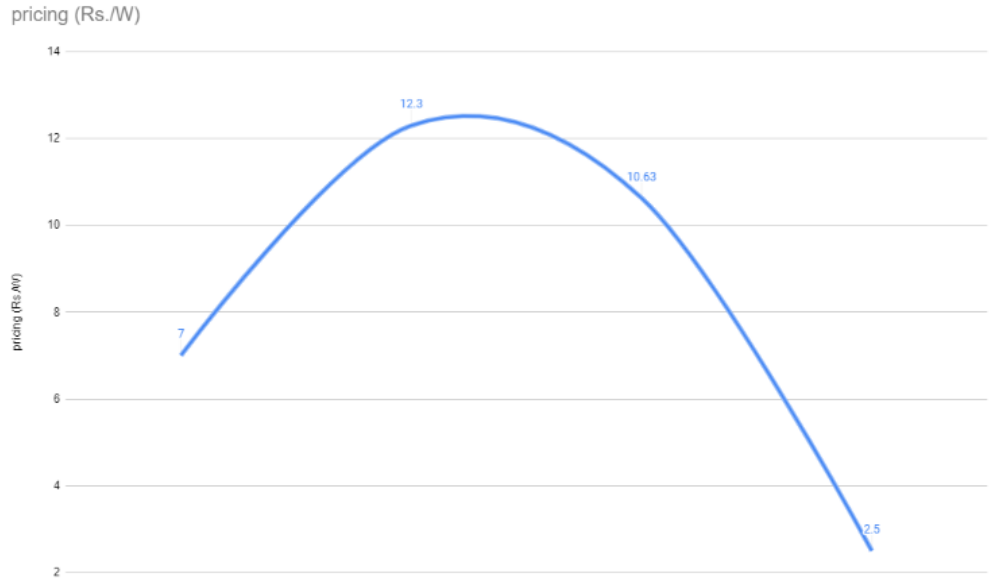


FIGURE 5.30: Pricing Scheme 2

Figure 5.30 shows the second pricing scheme, which peaks during the day and is lower in the early morning and the evening. This type of pricing scheme is particularly used in office areas where the amount of electricity usage peaks significantly during the morning to afternoon working hours. The sudden surge in the activation of many devices in offices usually leads to a huge surge in energy requirements. Hence it is enforced that the offices take the electricity consumption responsible for avoiding high bill charges. This pricing scheme is sampled from the Gaussian Normal distribution.

5.10.2 Monotonic Pricing Schemes

Next, we move on to the monotonic pricing schemes, where the pricing scheme monotonically moves in one direction. That means we get an increasing or decreasing bill charges trend over the time-of-the-day rating.

FIGURE 5.31: Pricing Scheme 3

We first look at Figure 5.31, which shows the first monotonic pricing scheme we will consider. This is a monotonically decreasing pricing scheme. This means that the

price of electricity is steep during the day time and low during the night time. This is primarily useful for restaurants and businesses that bloom at night time. With an anticipated energy demand, the electricity company prices minimum during the peak working hours and increases the prices during the off-hours when a high surge of electricity is not anticipated.

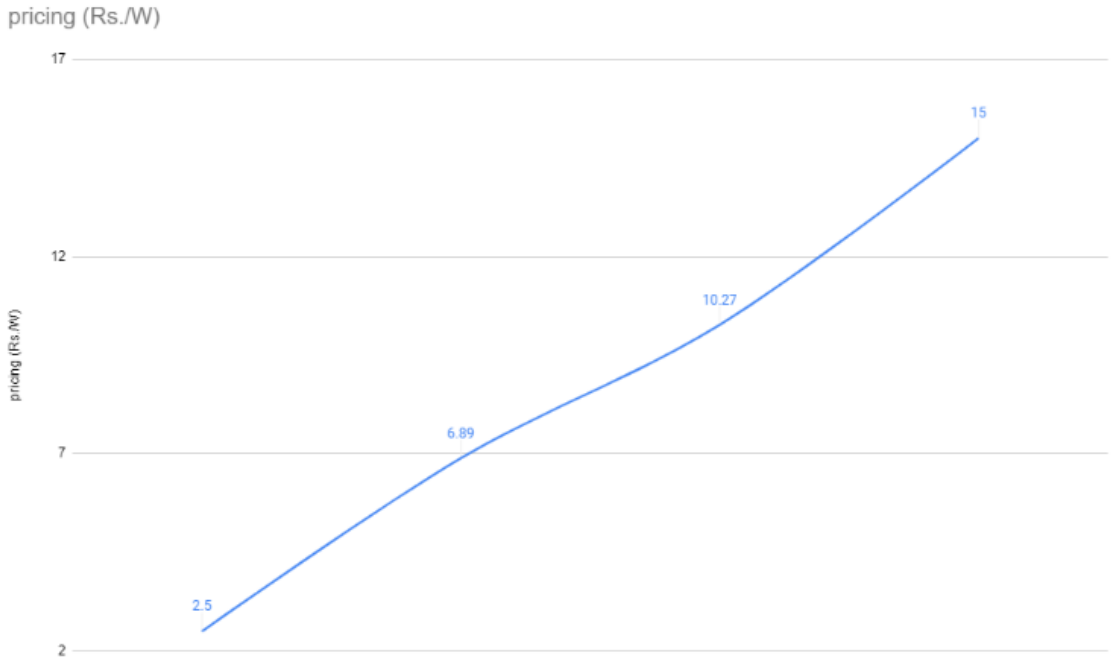


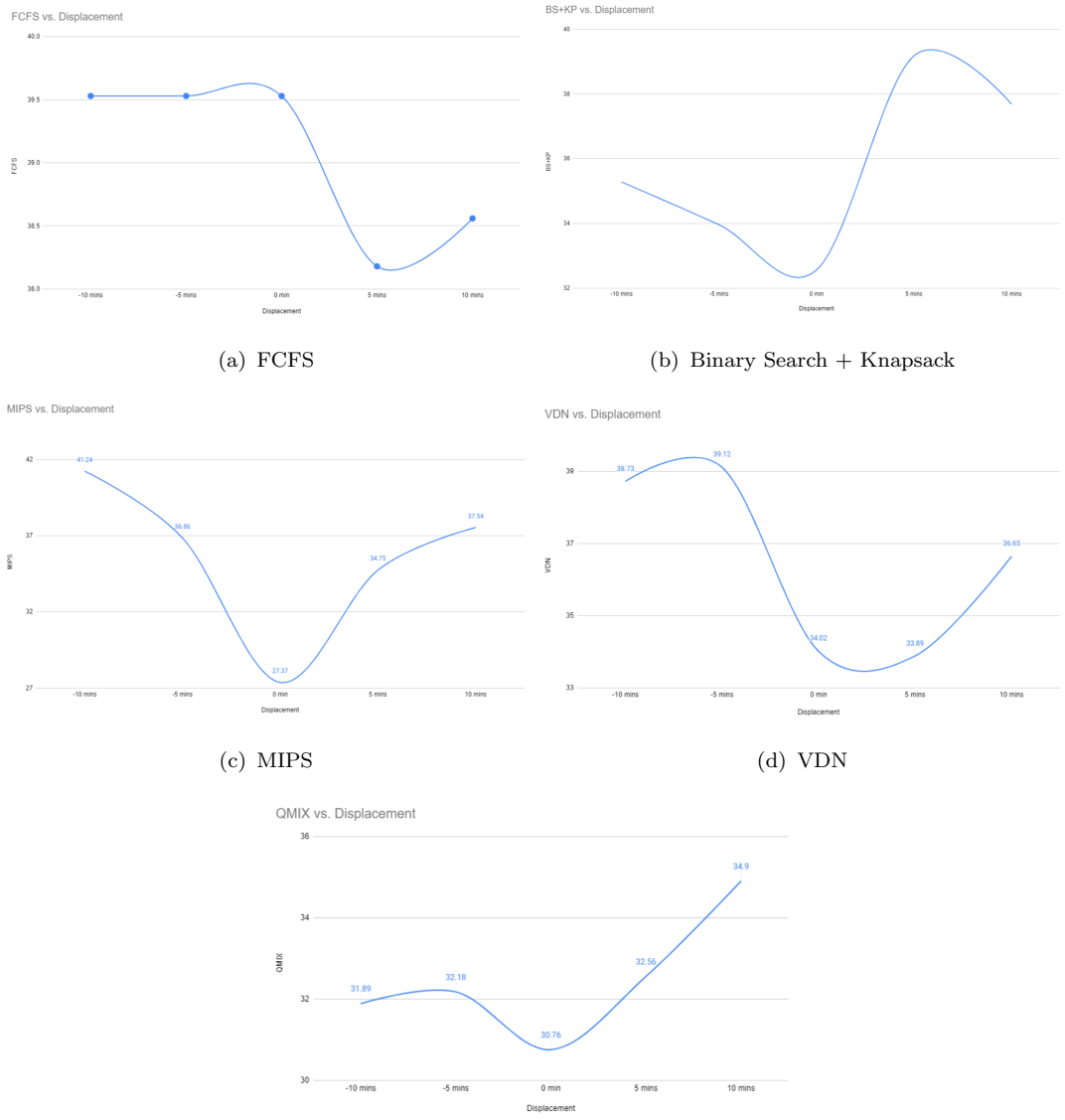
FIGURE 5.32: Pricing Scheme 4

Finally, we look at the last pricing scheme that we used for the experiments. The pricing scheme represented by Figure 5.32 is actually almost a mirror image of the previous monotonic pricing scheme represented by 5.31. This pricing scheme is a monotonically increasing pricing scheme; that is, the pricing is lower in the early morning and increases steadily towards the evening. This type of monotonic pricing scheme gives us an overview of a pricing scheme that penalizes a business due to high demand during the night time and is included purely to complete the general case scenarios.

For the following experiments, we will be using the schedule displacement technique. That means that once we get the generated schedule, we will iterate over all the jobs and randomly shift it by some number in the range $[1, \delta]$ minutes, where $\delta \epsilon \{-10, -5, 0, 5, 10\}$. A negative delay indicates that the job was done before the

time it was actually scheduled to do, and a positive delay indicates postponement of the job. The delay 0 is the original schedule.

5.10.2.1 Analysing the sensitivity for pricing scheme 1 5.29



(a) FCFS

(b) Binary Search + Knapsack

(c) MIPS

(d) VDN

(e) QMIX

FIGURE 5.33: Results for sensitivity analysis of pricing scheme 1 (Bill amount(in Rs. 1K) vs displacement)

First, we begin by analyzing the sensitivity of the model built on pricing scheme 1. From Figure 5.33(a), we can see that for FCFS scheduling, a preponement of the job

clearly has no effect, which is also expected. This is because the FCFS algorithm packs all the jobs in the earliest possible slot, which is free. Hence preponement and the schedule remain the same as the original schedule. However, we see that with the postponement of the schedule, there is a significant dip in the bill amount, which means there is a scope for improvement in the generated schedule. Figure 5.33(b) shows the results of sensitivity analysis for binary search-based optimization in combination with a knapsack solver. We see that the bill amount incurred increases both on preponing and postponing the schedule, which is also expected from a good schedule. The preponement of the schedule doesn't have much effect on changing the bill amount as compared to the postponement. This means postponing the schedule generated by the binary search-based method cannot be delayed in any case; however, doing the job somewhat earlier to avoid delay doesn't affect the schedule much. From the sensitivity analysis of the MIPS solver, as shown in Figure 5.33(c), we can infer that the schedule generated by MIPS, even though it provides us with the lowest bill amount, is very sensitive to any changes in the schedule. The user must schedule each job exactly on time or use automated help since a small delay of even 5 minutes can even be very costly.

Coming to the reinforcement learning algorithms, we analyze the results generated from **full training**. From the result of VDN shown in Figure 5.33(d), we see that the schedule generated is actually working quite in a mirror image fashion to the one generated by the binary search scheduler (Figure 5.33(b)). The schedule is highly sensitive to preponement as compared to the postponement. That is, doing a job early will have severe outcomes on the bill amount as compared to doing it a bit late. So the schedule generated by VDN is actually safe for delays. Finally coming to Figure 5.33(e) shows the sensitivity analysis of the QMIX model. We see that the QMIX model is sensitive to postponement as we see a steep rise in bill amount as we go towards the right. But suppose we see the magnitude of the effect of the postponement. In that case, we will see that it is very low compared to the effect of postponement on the previous model and hence is comparatively more postponement friendly compared to the deterministic counterparts, as we will see in the combined results in the next subsection. The schedule generated by QMIX supports preponing jobs since we don't see a significant effect on slight preponement.

5.10.3 Combined Sensitivity Analysis Results

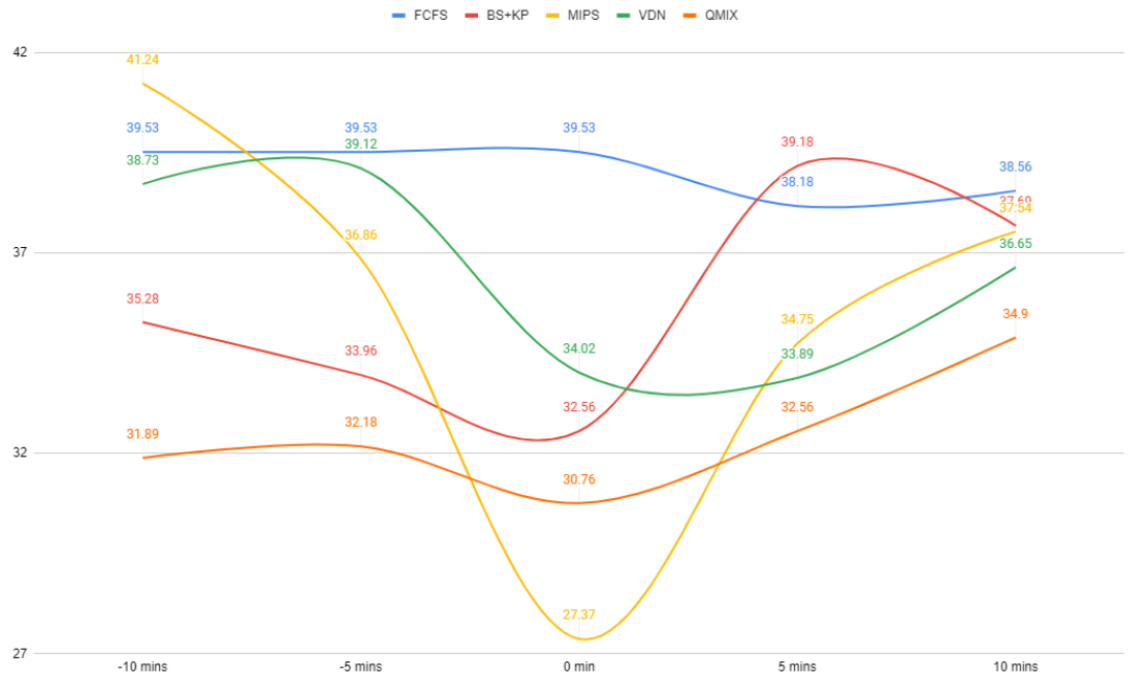


FIGURE 5.34: Combined Sensitivity Result for Pricing 1 5.29 (Bill Amount(in Rs.1k) vs displacement(in minutes)

We now combine the previous section's results to understand better the relative effect and the position these models stand on. We begin by analyzing the combined results of the models fine-tuned to work on the pricing 1 scenario as shown in 5.34. We see the MIPS perform extremely well in optimizing the bill amount, but the generated schedule is very sensitive to the time slot they have been arranged in. Shifting the generated schedule for multiple machines by 5-10 minutes has a detrimental effect on the schedule. As explained in the previous section, both QMIX and Binary search-based scheduling are sensitive to delayed scheduling. Still, as we can see from the combined plot, the schedule's effect is much less than the binary search combined with the knapsack method. Hence the delayed scheduling has more stability in the case of Reinforcement learning-based QMIX agents than deterministic counterparts.

Moving on to the result generated for the fine-tuning of the model on the pricing scheme-2 as shown in Figure 5.35, we see that the preponing and postponing cases of MIPs is still the worst case possible, but this is still the model which achieves the

least bill amount as compared to the other models. The QMIX model, in this case, outperforms the deterministic and other value decomposition networks. However, we see that we may achieve some incentives by preponing some of the jobs. This shows that there is still scope for improvement in the modeling. A similar kind of result is also observed in the case of binary search-based scheduling too. This also reinforces the fact that there is some similarity on the basis of the scheduling of both the algorithms, namely QMIX and binary search since both are based on the monotonicity fact and display some level of similar behaviors. VDN, as before, is lower on sensitivity to postponing as compared to preponing and the FCFS variant is invariant to preponing since early stacking and shows improvement on postponing.

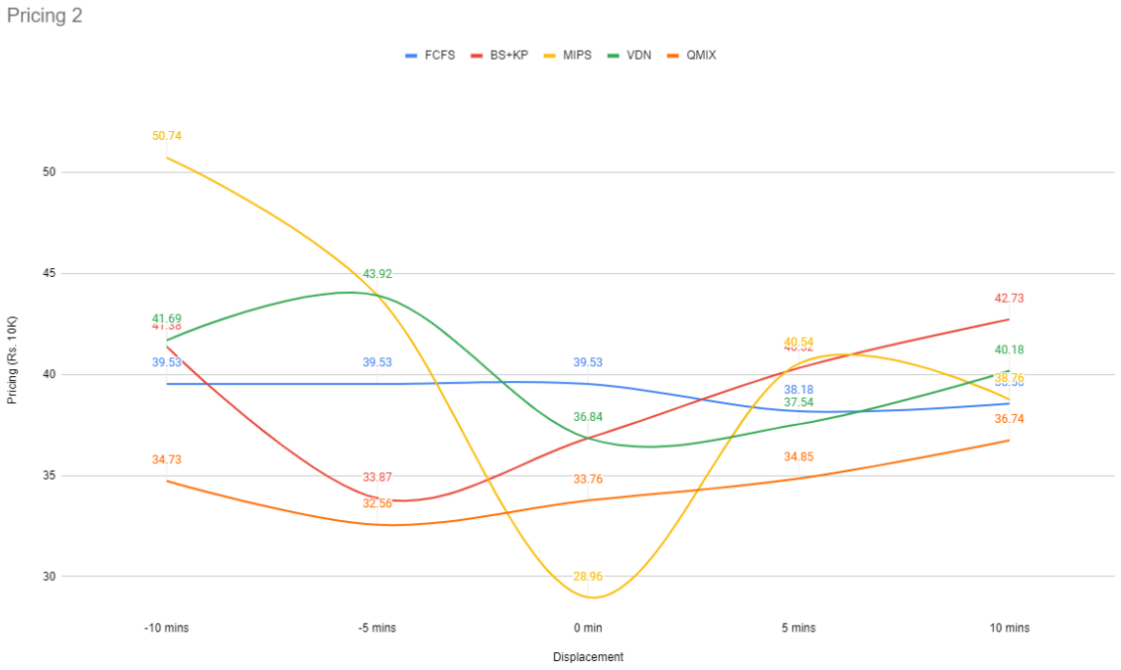


FIGURE 5.35: Combined Sensitivity Result for Pricing 2 5.30 (Bill Amount(in Rs.1k) vs displacement(in minutes))

Finally, we analyze the results of the monotonous pricing model too. Figure 5.36 shows the sensitivity analysis result of the pricing scheme-3 discussed above. Pricing scheme-3 is a monotonously decreasing model, which means the pricing is high in the early morning hours and lowers as we move to the late night hours. This fact is also reflected in the results. In most cases, we see that preponing the schedule severely affects the bill amount, but postponing the schedule showed some improvements in the bill amount. Since the MIP model had already stacked all the jobs towards the

lowest price part of the schedule, there is not much significant change in the bill amount resulting in postponing the job but a sharp increase in preponing it. For this pricing scheme, we see that the binary search model has improved performance compared to the qMIX model though nearly the same. The QMIX model, however, on extreme delay, had a better performance than the Binary Search-based scheduling option. However, overall we can see that most of the adaptive models learn to stack the jobs toward the end of the day for better billing. However, if the MIP model is run for all households or restaurants, then all the jobs will be strictly backed towards the end of the day, and there will suddenly be a huge surge in electricity. Comparatively, the scheduling generated by the Binary Search and QMIX algorithms are much more friendly to the power consumption and will not affect the peak power consumed much since there is quite an even distribution of energy even towards the end of the day.

FIGURE 5.36: Combined Sensitivity Result for Pricing 3 5.31 (Bill Amount(in Rs.1k) vs displacement(in minutes)

5.10.3.1 Analysing the sensitivity for Average Case

(a) FCFS

(b) Binary Search + Knapsack

(c) MIPS

(d) VDN

(e) QMIX

FIGURE 5.37: Results for sensitivity analysis of Average Case analysis (Bill amount(in Rs. 1K) vs displacement)

Now we come to the analysis of the individual model performance on the average case scenario as compared to the pricing scheme 1, as was discussed in the earlier sections. For each of the following averaged case results, we perform the scheduling 20 times, that is 5 times for each of the 4 schedules, and then take the average; since the reinforcement learning algorithms take uncertainty into account, they produce

different results every time, and we need to average over multiple generations to get the fair idea of the model's capacity. The results of the deterministic algorithm remain the same in all generations. Hence multiple generations in the same pricing scheme don't affect the averaging till the time we take an equal number of generations for each pricing scheme. Also, all these schedules are generated will full training; we will see the effect of incremental training in the next section.

For FCFS, from the results shown in Figure 5.37(a), we see that the average case scenario has nearly the same structure as the case with the pricing scheme -1. However, we see a higher average value due to its higher bill amount in the later cases. Figure 5.37(b) presents the result of averaging with the binary search combined with the knapsack algorithm. From this, we see an interesting observation that, on average case, the binary search has a lower effect on postponing the jobs as compared to postponing in the case of pricing scheme 1. This is partly because of the contributions of the monotonic models in the averaging too. Another interesting observation is the sensitivity to preponing the job increases. Thus, the resultant plot is now quite similar to that of the effects of the MIP model. Figure 5.37(c) shows the result of the average case analysis for the MIP model, and as expected, we see that the MIP model produces a very low bill amount but on disturbing the schedule, the penalty is usually very high, that is we can say that the schedule is quite unstable. Indeed we see that for the average case result, the preponing has a much more verse effect as we see a further increase in the steepness of the slope of the curve.

Coming to the results of the reinforcement learning models, we see that figure 5.37(d) shows the result of the VDN model on the average case scenario. We see that the VDN model, which was earlier only quite sensitive to preponing now becomes equally sensitive to both preponing and postponing the schedule. Still, the range over the sensitivity is reduced, which is a good sign as we know that preponing and postponing will not have severe effects as before. Figure 5.37(e) presents us with the result of the average case scenario with the QMIX algorithm. The QMIX shows a similar trend as most of the other models; that is, QMIX now has increased sensitivity on both preponing and postponing the schedule. But as we would have expected, the QMIX shows behavior quite similar to binary search-based scheduling due to its similarity in the monotonicity assumption; we see a reversal in the severity

effect. We see that QMIX now has lower severity on postponing as compared to preponing in the average case scenario.

5.10.4 Combined Average Sensitivity Analysis Anaysis

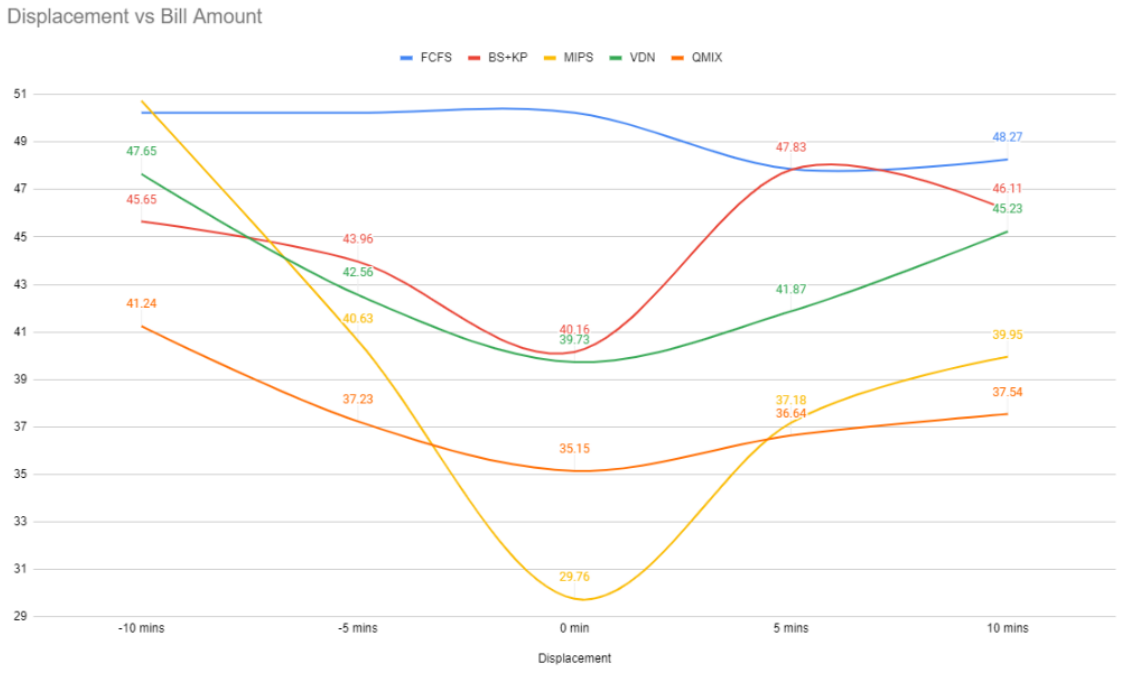


FIGURE 5.38: Full training - Combined Average Case Sensitivity

On analyzing the combined average analysis result we see that the MIP model still performs the best in terms of achieving the minimum bill amount and is very sensitive to any disturbance in the schedule as it has a detrimental effect on the bill amount so produced due to an error on the part of the human due to delaying or doing the chores early. The QMIX model is the best-performing model amongst the others in terms of achieving lower bill amount which is stable in terms of the effects due to human error like preponing or postponing the jobs. We see that the minimum of the VDN and the Binary Search algorithm is nearly the same when there is no disturbance in the schedule, but in case of a disturbance in the schedule, the VDN model has a lower effect as compared to the Binary Search algorithm. The FCFS algorithm as in general, has the worst performance.

Figure 5.39 shows the mean and variance of the model's performance in the average case scenario. We see that the mean value of the MIP is lower than any other model as can be seen from the combined plot of 5.38, but the variance of the MIP model is too high, near in the range of 60% as compared to the variance of the QMIX model which is near in the range of 10%. FCFS has the least amount of variance, which is also understandable due to the fact that there is not much possibility in terms of preponing, and postponing in the worst position in the day doesn't have much effect on postponing too. The VDN and the Binary Search-based algorithm have nearly the same variance and mean too in the average case scenario.

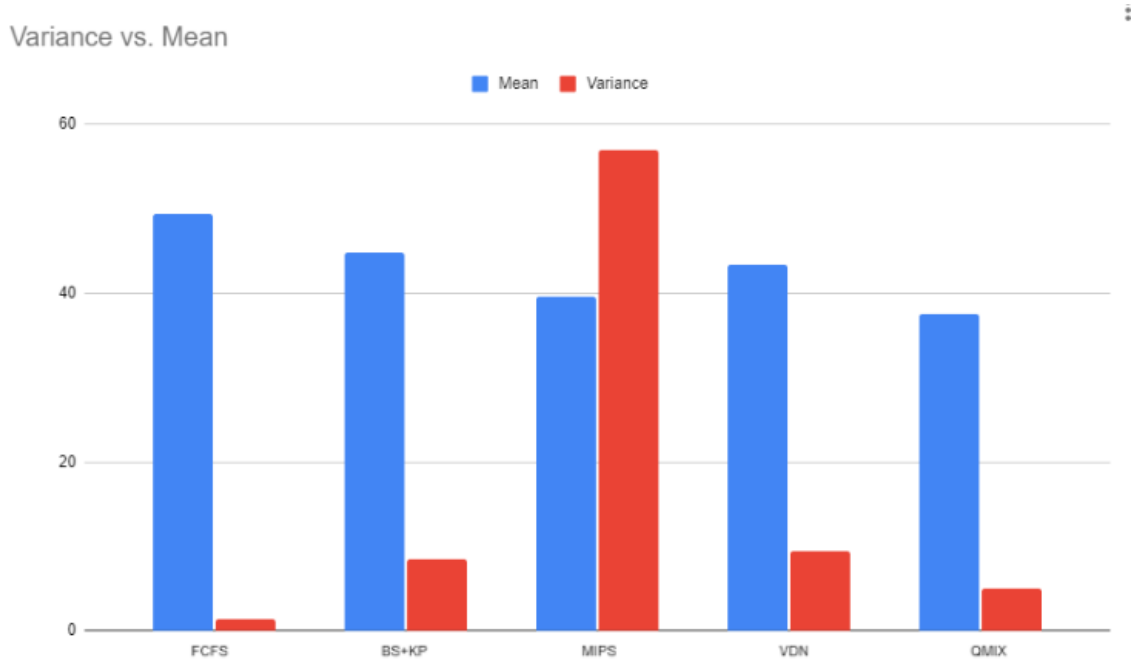


FIGURE 5.39: Full training - Combined Average Case Mean and Variance for different models

5.10.5 Introducing Predictor Network

We now introduce the predictor network-based modeling in conjunction with the QMIX model since it had the best result compared to the other models. Figure 5.40 shows the result of introducing the predictor network-based model. We see that the predictor network actually lowers the bill amount achieved by the QMIX model. The is some increase in the sensitivity on preponing and postponing, but as compared to

the previous QMIX model, we see a dip in the amount of penalty that the model was previously incurring. This means that the introduction of the predictor network into the modeling makes the generated schedule more stable. We see that postponing the schedule generated by the predictor network by less than 5 minutes has much less effective compared to that by shifting it by less than 10 minutes. However, in the case of the preponing of the schedule, that effect is reversed. That is, preponing by less than 5 minutes leads to an increase in the bill amount, but further preponing by 5 minutes leads to a drop in the bill amount, so generated with considering the disturbance.

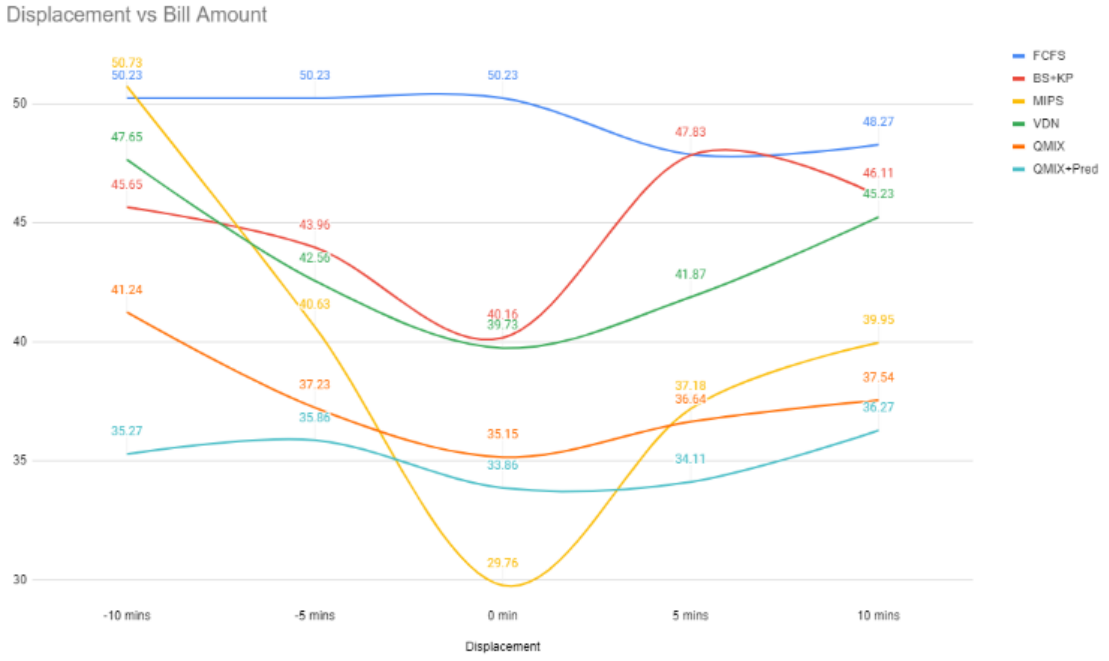


FIGURE 5.40: Result of Predictor network on Average Case Sensitivity

Figure 5.41 represents the man and variance results of the different models with full training on the multi-agent scenario after introducing the predictor network. We see that after the introduction of the predictor network, the effect of the variance due to any disturbance in the schedule is decreased further in comparison to the original QMIX model. We see an improvement in the variance from 3.63% to 1.12%, which is a near 0 variance, similar to the FCFS model. When used with the predictor network, we also see that the QMIX model performs much better in terms of the mean bill amount that it achieves in the different pricing models on average compared

to the MIP model. Even though MIP achieves a lower bill amount, the sensitivity to different displacements due to uncertainty leads to an increase in the mean bill amount that it achieves. On that front, the QMIX model performs quite better and further has improved results with the predictor network with mean reducing from Rs. 39.65K to Rs. 37K.

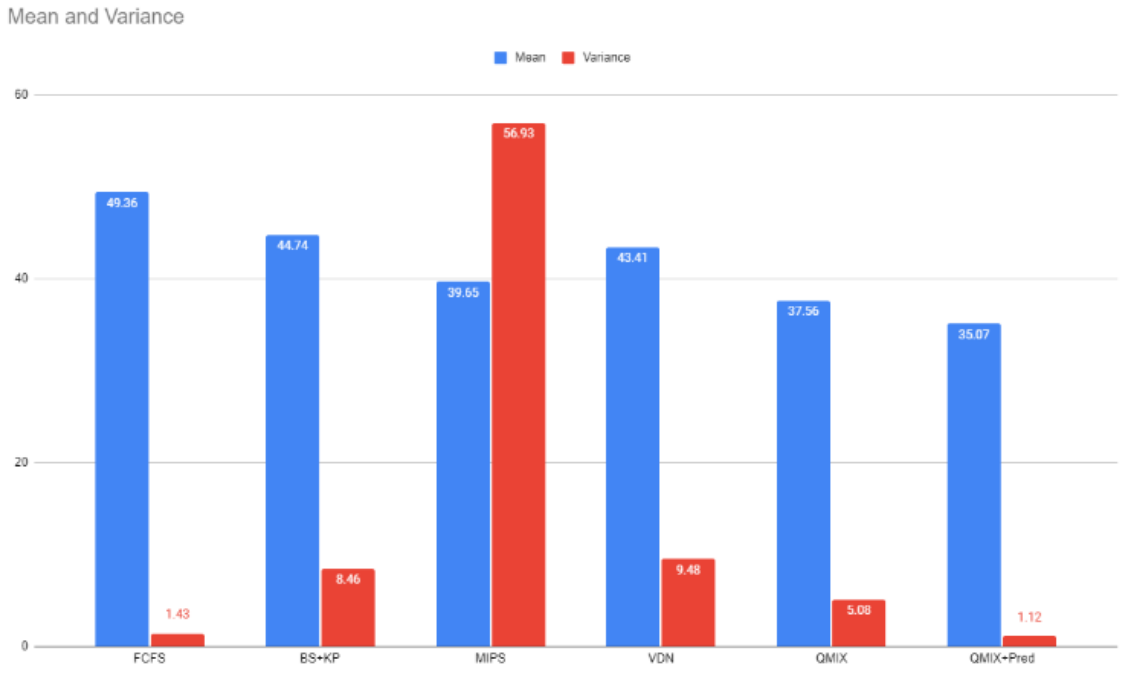


FIGURE 5.41: Result of Predictor network on Mean and Variance

5.10.6 Improvements with using Incremental Training

Now we make use of incremental training for training the models and then study the results obtained. The incremental training is based on the idea of reusing the knowledge of the model acquired for learning a smaller set of items and extending it for a larger set of items as discussed in 4. Figure 5.42 shows the result of the combined analysis report on the sensitivity of the models trained with incremental training. From the results, we see that the incrementally trained models have a bit higher sensitivity to the displacement effects due to human uncertainty as compared to the full training of the same models. We additionally observe the most significant fact that using the incremental training methodology also lowers the minimum bill amount achieved further, and we have achieved bill amounts very close to that

achieved by the MIP solver, but the schedule generated by the reinforcement learning algorithms is much more stable compared to the deterministic counterpart of MIP. Deterministic models do not need any training. Hence there is no incremental training for the deterministic models. Interestingly, the incrementally trained VDN model with predictor network, when shifted by less than 10 minutes prior, that is, randomly preponing the jobs by less than 10 minutes, has lower bill amount consumed as compared to the QMIX model trained under the same environment.

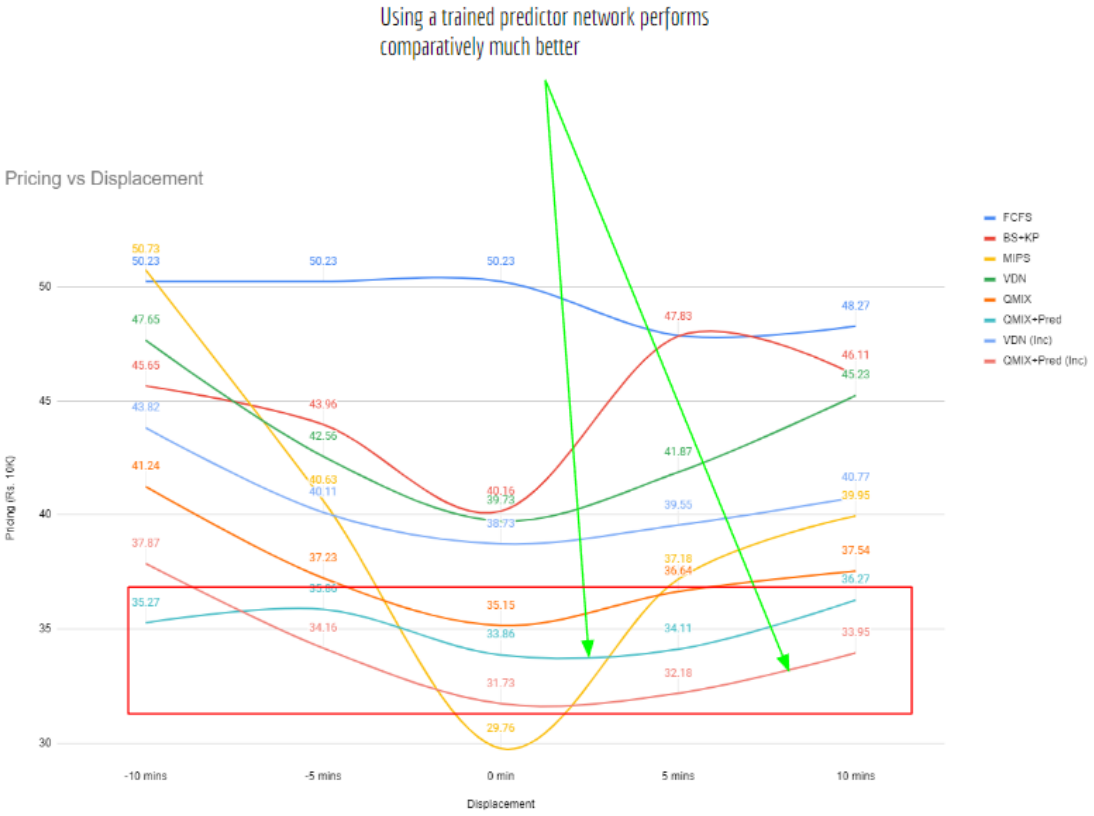


FIGURE 5.42: Incremental Training - Combined Average Case Sensitivity

Figure 5.43 presents us with the results of the mean and average obtained with different models on incrementally training them compared with the full training methodology. Clearly the average of bill amount achieved in the different scenarios for incrementally training the model is lower when compared to the full training, but we see an increase in the variance value for incrementally trained models. This is primarily because when the full training is performed, it takes into account all the effects of all the models over all time steps, but the incremental model is focused

on optimizing one subset of items at a time, hence suffers from higher variance comparatively. Thus we can conclude that with full training, we have a higher average bill amount but lower variance, but with incremental training, we have a lower bill amount but higher variance.

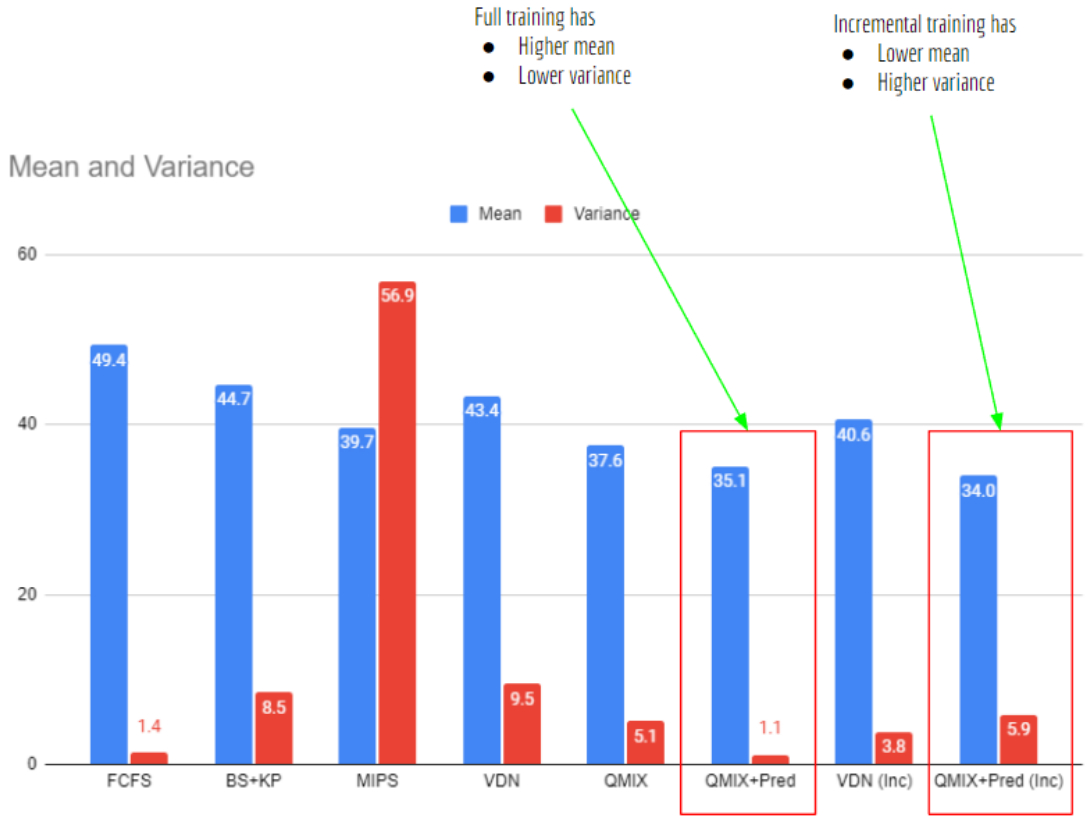
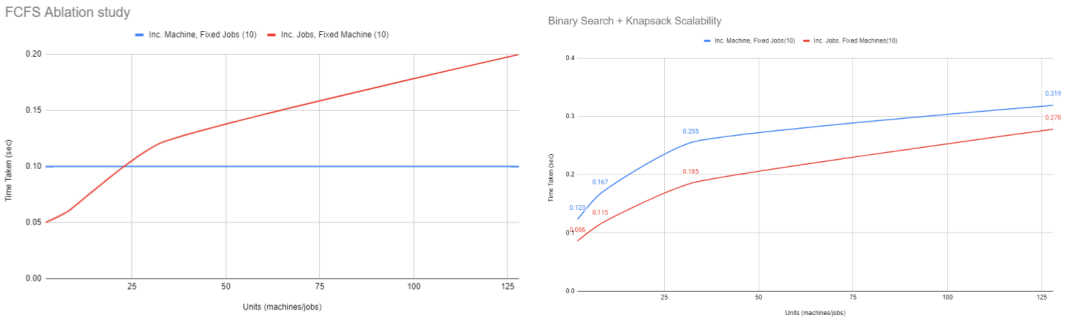


FIGURE 5.43: Incremental training - Combined Average Case Mean and Variance for different models

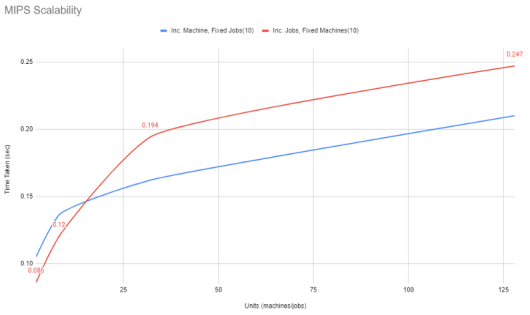
5.11 Scalability of the Models

The following figures 5.44 and a5.45 represent the result of the scalability of the models in terms of time taken. For the deterministic models, as shown in figure 5.44, we see the time for producing the schedule for each algorithm. While in the case of the reinforcement learning algorithms, we see the time for training the model which is significant as the number of machines for which the scheduling needs to be learned grows.



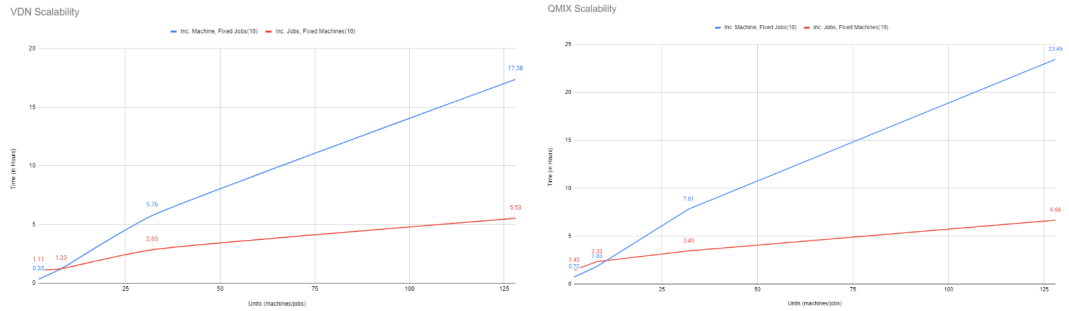
(a) FCFS

(b) Binary Search + Knapsack



(c) MIPS

FIGURE 5.44: time taken inferencing with deterministic models



(a) VDN

(b) QMIX

FIGURE 5.45: time taken incrementally raining the reinforcement learning models

Figure 5.44(a) represents the result of the inferencing of the FCFS algorithm, which is constant if we increase the machines keeping the jobs constant and increasing the number of machines as all the jobs will be equally distributed among the machines. Still, if the jobs are increased keeping the machine constant we see a slight increase in the inferencing time due to scheduling more jobs. Figure 5.44(a) represents a

similar result with the Binary search scheduling algorithm. As the jobs increase, the algorithm needs to iterate and transform more jobs to see a perfect location. The complexity of the knapsack also increases as the number of items increases. Hence we see a steady increase in the time taken for inferencing. As the number of machines increases, only the complexity of the knapsack increases, but not significantly due to the fact that the jobs are distributed equally on the machines, so just the knapsack machine is run more time on a smaller number of jobs per machine. Figure 5.44(b) shows that MIPS is also affected both on the fronts of increasing machines and increasing jobs. There are no significant constraints between the machines in the MIP modeling. As the number of machines increases, there are more possible values that the jobs can take (due to equal distribution), and looping over the number of machines takes a bit of time, hence a slight increase in time with more machines. As the jobs increase, the number of constraints added also increases proportionately. Hence the time taken for the solver increases significantly. VDN uses incremental training methodology for training, thus, the major effect on the training time is due to increasing the number of machines, as seen from figure 5.45(a). This is because the incrementally is based on the increasing number of machines based on the knowledge gained from a smaller subset of machines. Since processes are equally distributed among the available machines, increasing the number of processes primarily affects the processing time. QMIX also follows a similar training technique, due to which we see similar trends as compared to VDN as seen from 5.45(b). The time taken for training the QMIX is quite higher as compared to the VDN primarily because of the more computations involved in QMIX.

5.12 Advantage of Reinforcement Learning over heuristic bounds

- RL Agents can efficiently model the best scheduling considering the human uncertainties during run time, like delayed scheduling with bounds. This can be easily verified during real-time experience since while training, this was considered while modeling the environment.

- Reinforcement Learning-based approach can efficiently arrange the jobs within the day by looking at different time points. This has been proven and explained in the case of a comparative study between binary search and QMIX scheduling results.

- Reinforcement Learning-based approach is highly scalable as compared to heuristics. There is no dependence on the number of jobs (as explained in the modeling). Hence, any number of jobs can be added, and the model can still schedule them dynamically and efficiently during run time with less inference time. However, the heuristic approaches are limited by the resources like the memory available for computation. Since they are solving the NP-hard problems with larger constraints, the algorithms start going slower.

- From the results of the single agent, we observe that RL Agents can model the characteristic stochastic quantities like the satisfaction of the user much better as compared to the deterministic counterparts, and this quality is also extended to the multi-agent scenarios too.

- With the aid of a predictor network we can achieve results much close to heuristic lower bounds and much more stable than the heuristic schedules in terms of their sensitivity to human uncertainties.

5.13 Results of Genetic Algorithm Solver

Figure 5.46 presents us the result of the genetic algorithm, which is plotted at specific points of the iteration, that is, after every $2^k \times 100^{th}$ generations. In each generation, we divide all the offspring bill amounts into 3 buckets and use the average to determine the circle center. These buckets have an equal number of offspring. Since each generation has 100 offspring, thus each bucket will have 333 offspring on approx. Averaging gives us an idea of the trend of that particular bucket, and bucketing is similar to binning, which helps us to understand the direction of the concentration of the offspring. Buckets are built after sorting all the offspring by their fitness value, where fitness incorporates the bill amount. Bucket 1 hence will always have a lower bill amount, followed by Bucket 2 and finally Bucket 3. From figure

5.46, we see that the results of all the buckets improve over different generations. The concentrations of the buckets, however, shift over different generations. In the early generations, Bucket 1 and Bucket 2 are quite close together, but Bucket 3 is quite far. However, in the later generations, we see that Bucket 2 shifts closer to Bucket 3 and Bucket 1 are further apart, which means that the concentration of the offspring shifts towards the higher bill amount side. This means that most of the offspring are not improving after a certain number of generations and may contribute due to the fact that direct crossover of two healthy parents may introduce non-healthy offspring. Comparing the performance of the genetic algorithm with the above-analyzed solvers like QMIX and MIP, we see that Genetic Algorithm doesn't perform at the same level of efficiency. MIP was the most efficient, with a bill amount o Rs. 29.76K, and Incrementally Trained QMIX with Predictor Network had achieved a bill amount of Rs. 31.73K, which was more stable, but the bill amount achieved with Genetic Algorithm is in Rs. 38K ranges on average, and the best schedule has a bill amount of Rs. 37K even after 1600 generations. We maintain multiple offspring of each generation primarily because the offspring with the lowest bill amount is not always the schedule that is most stable. Hence variety helps us make informed decisions about the best schedule for different use cases.

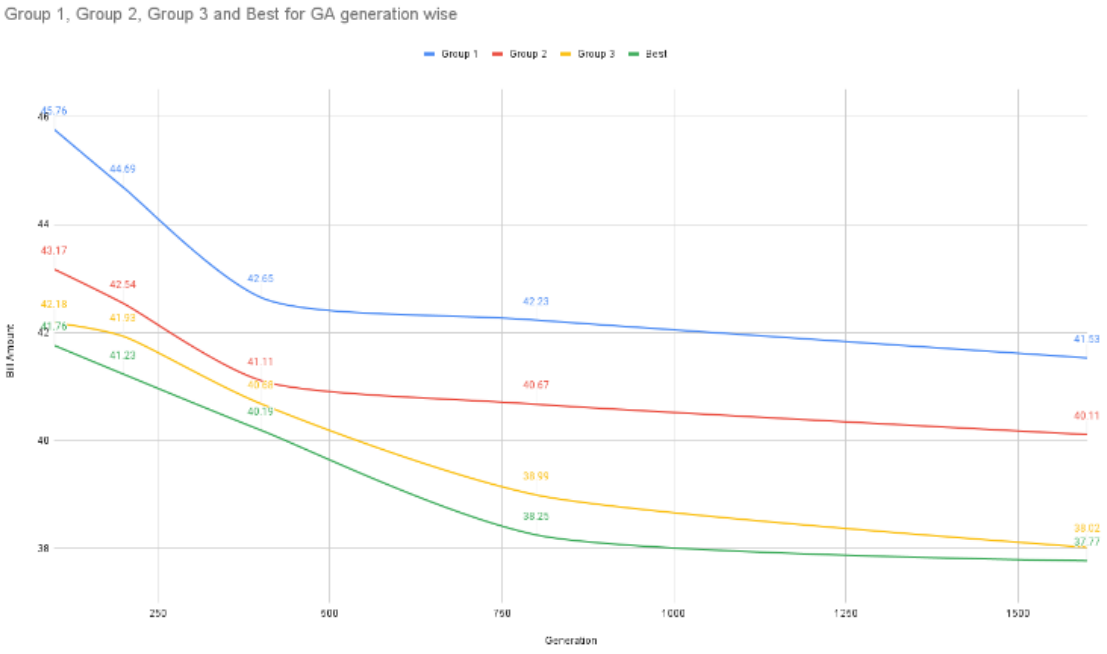
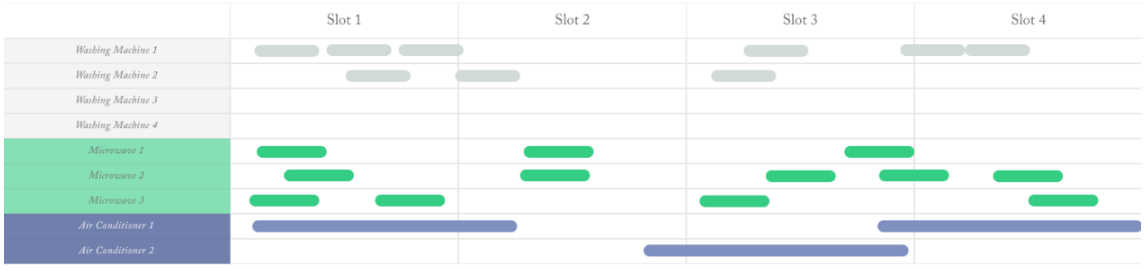


FIGURE 5.46: Result of the generation-wise performance of the Genetic Algorithm solver for the problem

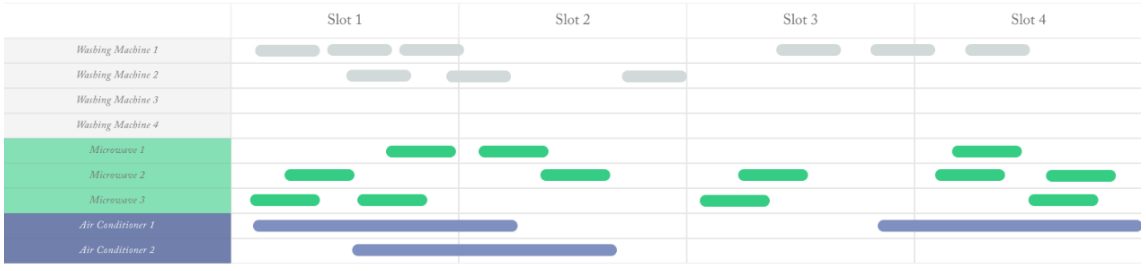
5.13.1 Analysis of Schedule Generated

(a) Schedule 1

(b) Schedule 2



(c) Schedule 3



(d) Schedule 4

FIGURE 5.47: Examples of different schedules after 3200 generations picked randomly from group 1

Figure 5.47 shows the 4 randomly sampled examples of the schedules generated after 3200 generations. From the schedule 5.48, which represents the scheduling with the best bill amount, we can see that the algorithm learns over generations that stacking the jobs in the lower bill amount is more efficient while, at the same time, it is trying to maintain a lower peak power while distributing the jobs over the day. From the previously analyzed schedules of the Binary Search Algorithm and QMIX issues with stacking on the early part of the day (less exploration on later halves), we see that the genetic algorithm has much better exploration of the entire day as it is able to spread jobs over the day. This is primarily due to random initialization of the jobs.

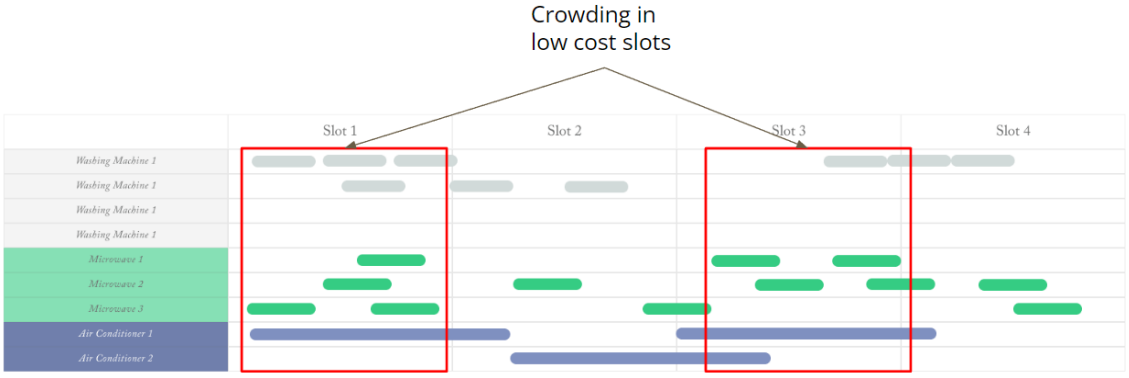


FIGURE 5.48: Analyzing schedule generated Genetic Algorithm

5.13.2 Sensitivity Analysis

Figure 5.49, which presents the result of the sensitivity analysis of the genetic learning algorithm, shows that the average sensitivity of the group 1 offspring is much higher compared to the average sensitivity of the group 2 element and group 3 elements. The average sensitivity of the Group-2 and Group-3 elements is much closer as compared to Group-1. By Group, we here refer to the different Buckets described above. The group 3 elements, though, have a poor performance and have the least sensitivity among all other elements as we see that displacement of the schedule generated has very little effect. This may be due to the fact that the schedule generated is not much good, and shifting the elements in their current region have comparatively less effect than moving out of strongly packed low-pricing regions. We can extend our study to include the features that make group-3 more stable as

compared to group-1 so that we can incorporate those features in group-1 to make group-1 more stable.

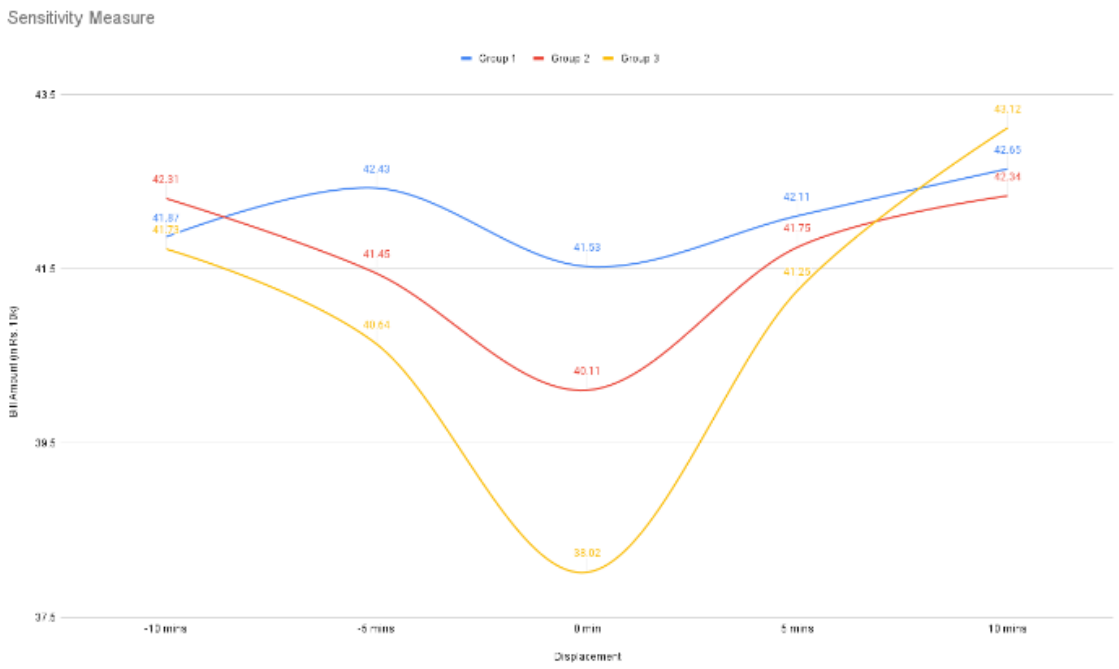


FIGURE 5.49: Sensitivity analysis of Genetic Algorithm

5.13.3 Time and Memory required for generations

From figure 5.50, we see that time taken for generation each generation of the genetic algorithm is nearly constant. This is inferred from the fact that the total time taken for generation i^{th} generation is nearly linear in the number of generations produced, that is, i. For producing 1600 generations wee se that we need approximately 811 minutes which is nearly 13 hours 30 mins, significant time, comparable to the time taken to train the reinforcement learning agents. For computing the memory consumed, we need to consider the following items:

- 1000 offsprings are present in each generation.
- Each chromosome is of length 200.
- Each unit of the 200-part chromosome is of 500 Bytes structure containing the data for scheduling the job

Thus the total memory consumed in our case is $(1000 \times 200 \times 500) * 2 = 200MB$ per generation of the genetic algorithm, which is also in consensus with the memory consumption detected on real-time algorithm execution. Here we multiply by 2 for the current and the next generation when being produced in memory simultaneously for selecting the healthy offspring.

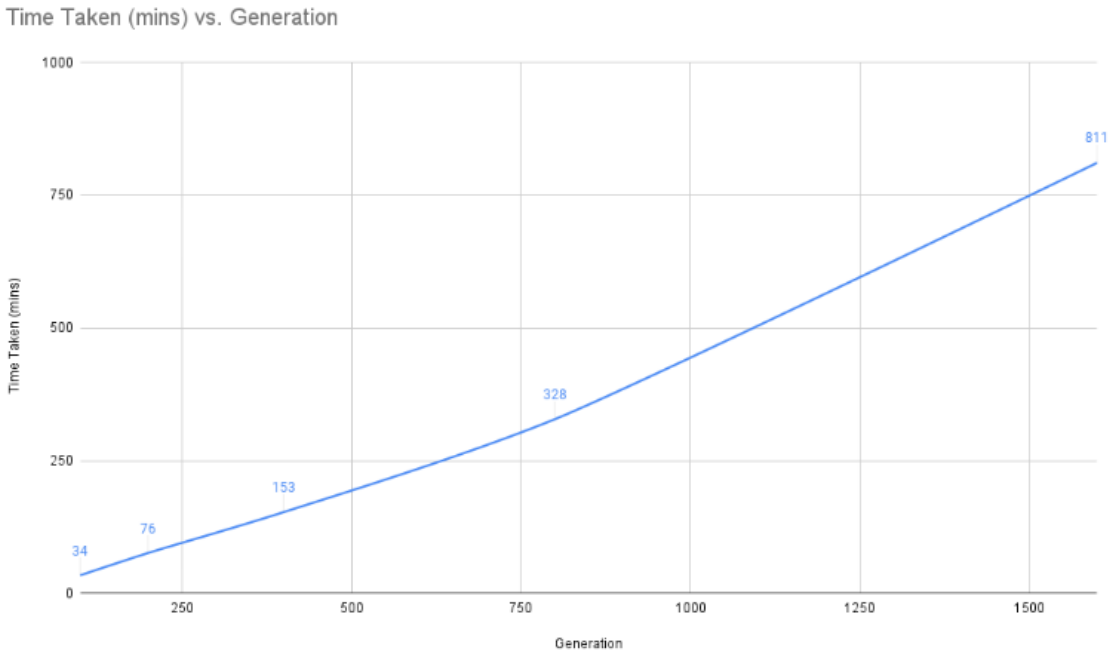


FIGURE 5.50: time of generating the generations

5.13.4 Effect of Reducing Population Size

From the following figure 5.51 and 5.52, we see the result of the performance of the genetic algorithm on reducing the population size from 100 to 500, that is, reducing the diversity. Figure 5.51 shows that on reducing the population size initially, the offsprings suffer much due to decreased diversity. We cannot attain the schedule with the best performance on a similar generation with 1000 offspring. However, due to the reduced number of offspring per generation, there is lower computation and memory consumption, so we have the bandwidth to generate more number of generations. We see that after 3200 generations, the 500 population is able to achieve performance a bit better compared to the 100 offspring algorithm. Figure 5.51 shows that the time taken for 1000 offspring for 1600 generations is nearly the

same as that of 500 offspring for 3200 generations. See that the result of Figure 5.51 is shown in a logarithmic scale.

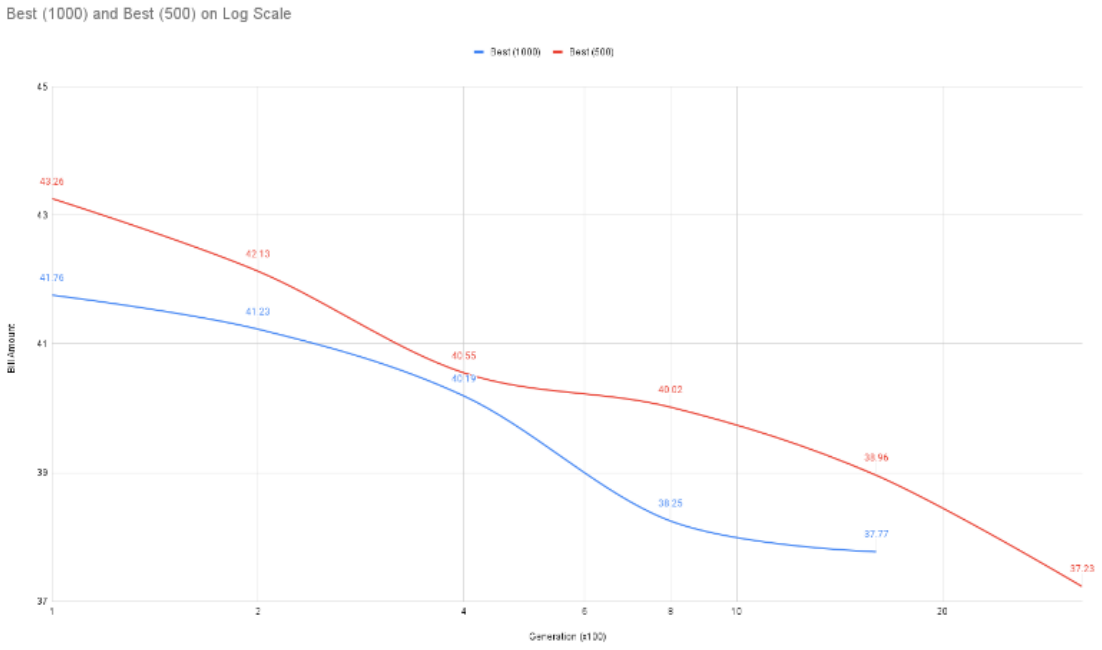


FIGURE 5.51: Result of performance on reducing population size

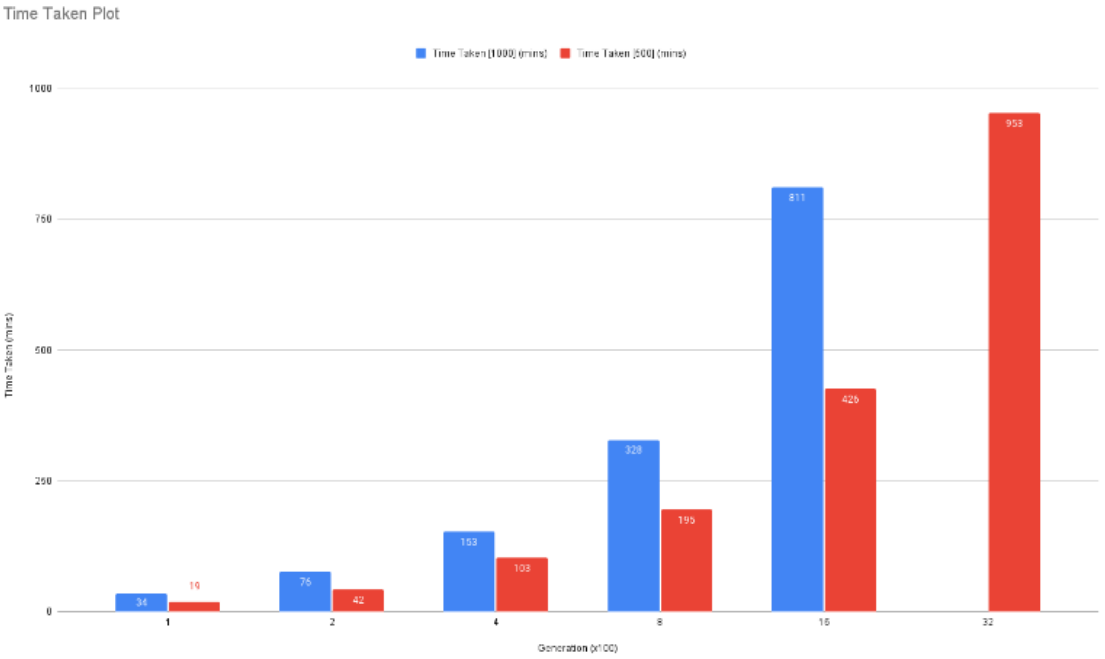


FIGURE 5.52: Result of time taken on reducing population size

5.13.5 Diversity Analysis on Genetic Algorithm Populations

5.13.5.1 Genotypic diversity

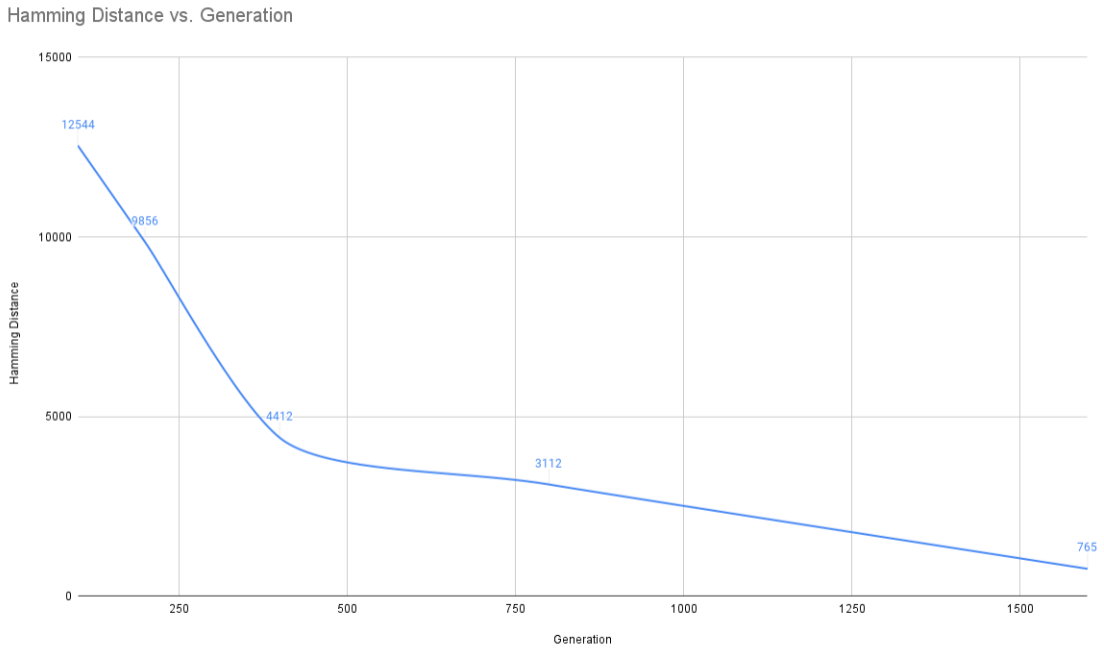


FIGURE 5.53: Result of Genotypic Diversity Computation

For presenting the results of the genotypic diversity, we had the following two options as described in the methodology section

- Hamiing Distance
- Sharon Entropy

We make use of the hamming distance to show our results for the genotypic diversity of the population in each generation of the genetic algorithm. Figure 5.53 shows the results of the diversity computation for the population size of 1000.

- We see a steady decrease in the population's genotypic diversity as the generations progress.

- The decrease in the population size is quite steep during the first few 500 generations, then the rate of the decrease reduces significantly.
- This may be due to the fact of getting stuck in a local minimum or reaching the valley of the global minimum.
- The decrease in genotypic diversity raises concern that our genotypic variation exploring capacity is reducing over time with the generations.

5.13.5.2 Phenotypic diversity

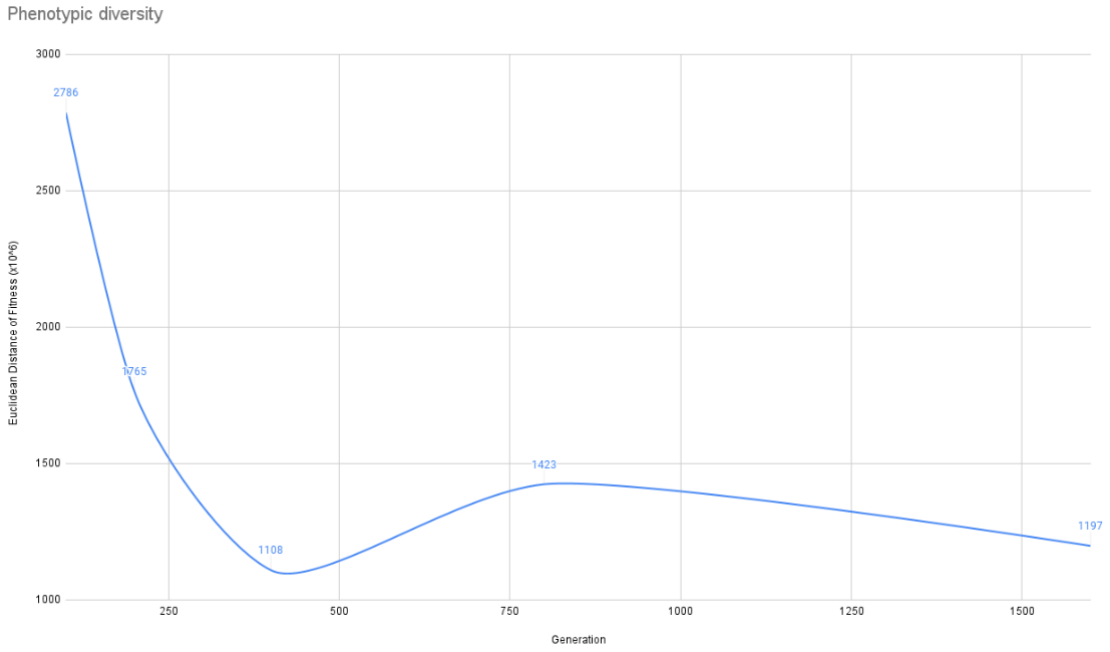


FIGURE 5.54: Result of Phenotypic Diversity Computation

- For computing the phenotypic diversity, too, we had multiple options: use either Euclidean distance of the variance or standard deviation.
- We observe that the phenotypic diversity initially declines steeply, but after around 450 generations, we see an increase in the phenotypic diversity, and after around 800 generations, it again keeps decreasing.
- From 450 to 800 generations, getting into a new valley may lead to an increase in the phenotypic diversity of the available population.

- The decrease in phenotypic diversity raises concern that our phenotypic variation exploring capacity is reducing over time with the generations.

5.13.5.3 Species Count

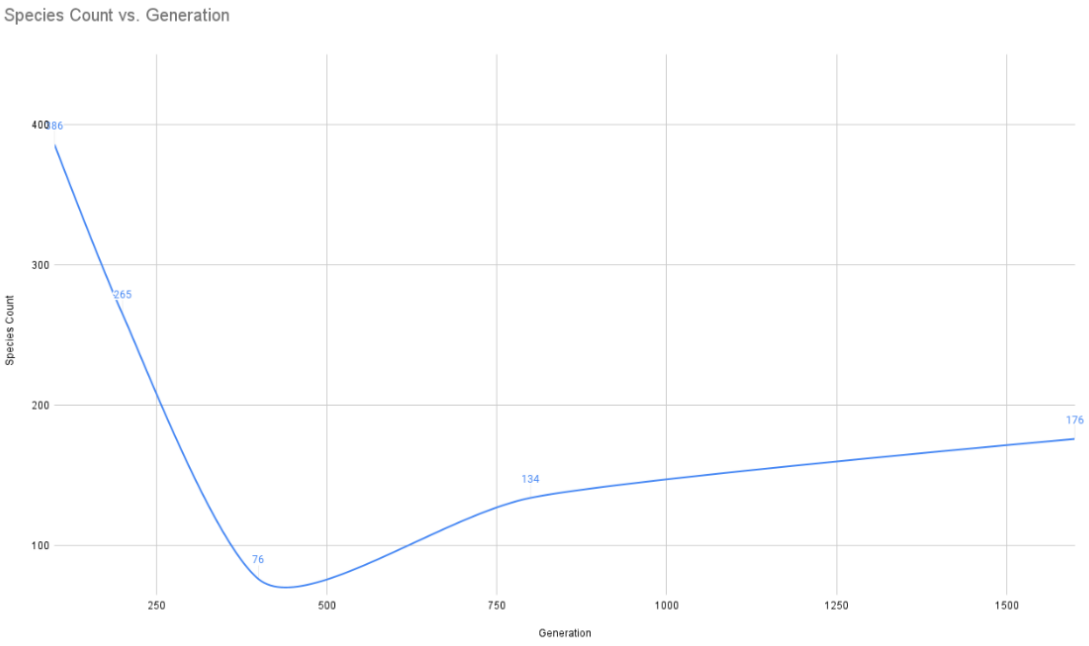


FIGURE 5.55: Result of Species Count Computation

- We run a DBSCAN - Density-based clustering algorithm using the Euclidean distance as the metric of distance and fine-tuned EPS (Distance Bound). Since the clustering is density-based it clubs together the genets that have a commonality in their evolutions and hence gives us an idea of the species count for the current generation.
- We see that the species count initially decreases, but later it increases after about 450 generations.
- This means that the evolutionary similarity among the chromosomes again starts reducing after a certain number of generations, and thus, we get a good variety accordingly.
- We used **sklearn.cluster.DBSCAN** for performing the above-mentioned clustering.

5.13.5.4 Fitness diversity

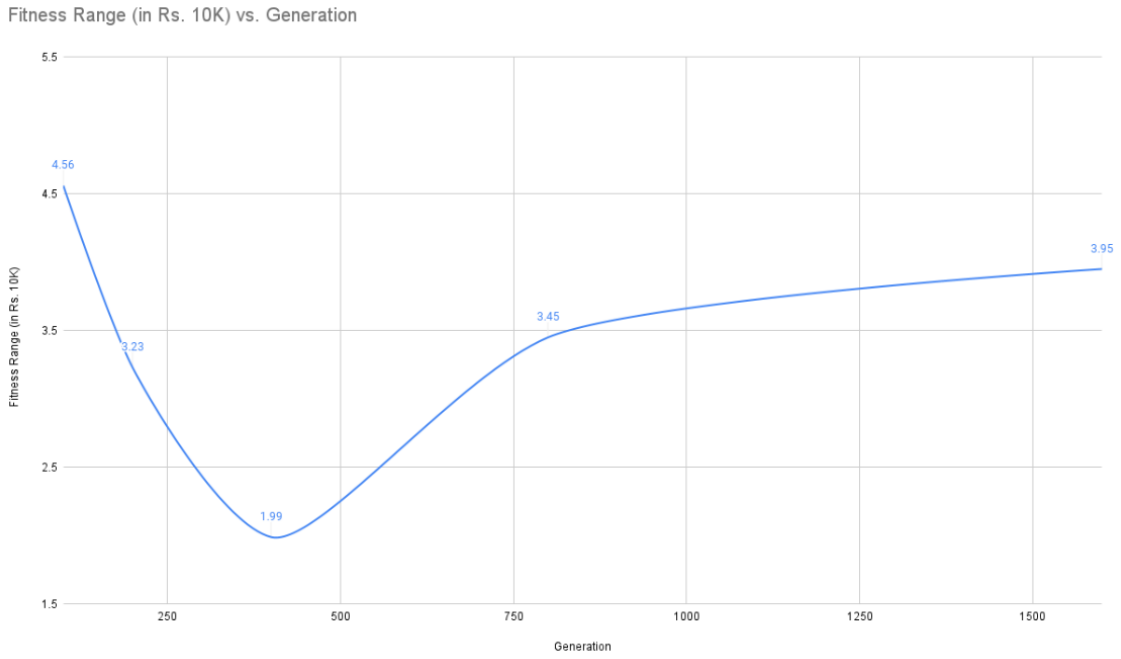


FIGURE 5.56: Result of Fitness Diversity Computation

- We choose to calculate the Fitness range among the chromosomes of that generation to get an idea of the fitness diversity among the chromosomes.
- We see that the fitness range starts with a pretty good range for diversity initially and then falls steeply for the first few generations up to 450 generations. After 450 generations, we again see a rise in the fitness range which is much steeper, and after 800 generations, the slope of the increase reduces, showing that the variety of fitness increases again after the initial reduction.

We present 4 different types of analysis for the genetic diversity of the algorithm. Each of the above metrics shows a different feature of the population and hence has a different interpretation of the result generated. As for Phenotypic and genotypic diversity, the reduction in diversity is a serious concern as we see a steep, followed by a steady decline in diversity. However, with fitness diversity and species count, we see an increase in diversity after some number of generations. The best result will be for all the metrics of diversity to be at least constant or increase over time for better exploration.

Chapter 6

Future Work

6.1 Using a Renewable Source along with the Grid

- Instead of making use of the power from the Grid as the sole source of energy, we will also make use of the renewable sources of energy like solar power and Hydro power and store the power supplied from them in the battery first.
- Storing the energy in the battery is important since the power from renewable source are not steady and users need steady power supply.
- Once the energy is stored in the battery, we can use the Reinforcement based schedule to optimize the objectives by balancing between both the grid power and the battery power.
- Reinforcement learning will help us train model which will help us utilize environment friendly alternatives to energy generation very efficiently

6.2 Generation Side Scheduler

- Extend the working of the scheduler to support not only the demand side but also the supplier/generation side.
- That is the RL based Scheduler will also schedule as to how the power generation should be managed between different generators so that the cost of generation is minimized while still meeting Demand side needs.
- Managing generation of electricity is also a similar problem in which we schedule the generators instead of the household devices. Having RL agent on both front of the consumer and the provider will reduce cost of generation as well as consumption. Thus the hybrid system will be very beneficial.

- Consecutive iterations of the brownouts may also significantly hamper the devices, even with recovery setup, hence scheduling the brownout for not hampering the device while recovery is also an important area to work on.

6.3 Scheduling Brownouts

- A brownout is an intentional or unintentional drop in voltage in an electrical power supply system. Intentional brownouts are used for load reduction in an emergency. The reduction lasts for minutes or hours, as opposed to short-term voltage sag (or dip). The term brownout comes from the dimming experienced by lighting when the voltage sags. Different types of electrical apparatus will react in different ways to a sag. Some devices will be severely affected, while others may not be affected at all.
- Different levels of brownouts may include allowing only 1 fan or one light or only essential devices to be run. Determining what level of brownout will not hamper the users and their devices significantly. Also since the intentional brownouts are done primarily by the electricity companies in sub-urban areas for cost cutting, simultaneously, the agent must also maximize the profit while doing so.

6.4 Extending Incremental Training

- Using the incremental training method described in Chapter 5 we can extend it to testing for power based increments. For example, number of machine on which the training in happening increments as $2^0, 2^1, 2^2, ...$
- Instead of Adding a new layer for training for each increments, we can compress the number of layers require using a binary fashion counter. Like, if we are training for machine 11 (binary 1011), then we will train layer 1, 3, 4 (corresponding to set bits) and freeze rest layers.
- We can also model the incremental training system as LSTM network instead of using 2^N action space. The motivation behind this is that the devices are identical and we need to schedule multiple units of the same device, so we can make a neural network and add feedback layers while maintaining long term dependencies using LSTM, and schedule each device sequentially.

Bibliography

Agency, European Environment (2018). "Environmental indicator report 2018," in: *European Union, Tech. Rep. ISSN 1977-8449*. URL: https://www.eea.europa.eu/airs/2018.

Chavali, P., P. Yang, and A. Nehorai (2014). "A Distributed Algorithm of Appliance Scheduling for Home Energy Management System". In: *IEEE Transactions on Smart Grid* 5.1, pp. 282–290. DOI: 10.1109/TSG.2013.2291003.

Ehsanfar, Abbas and Babak Heydari (2018). "An Incentive-Compatible Scheme for Electricity Cooperatives: An Axiomatic Approach". In: *IEEE Transactions on Smart Grid* 9.2, pp. 1416–1424. DOI: 10.1109/TSG.2016.2591507.

Gholian A., Mohsenian-Rad (2016). "Optimal Industrial Load Control in Smart Grid. IEEE Transactions on Smart Grid". In: *IEEE Transactions on Smart Grid* 7.5, pp. 2305–2316. DOI: http://dx.doi.org/10.1109/TSG.2015.2468577. URL: https://escholarship.org/uc/item/59v9g0h9.

K Raheja Group, Power (2018). "KNOW YOUR BILLS". In: URL: https://www.krahejacorppower.com/know-your-bill.

Kahan, Ari (2019). "The Middle East, Africa, and Asia now drive nearly all global energy consumption growth". In: *U.S. Energy Information Administration, Tech., Rep.* URL: https://www.eia.gov/todayinenergy/detail.php?id=37932.

Kim, Seungchan et al. (Aug. 2019). "DeepMellow: Removing the Need for a Target Network in Deep Q-Learning". In: pp. 2733–2739. DOI: 10.24963/ijcai.2019/379.

Klemenjak, Christoph et al. (Apr. 2020). "A synthetic energy dataset for non-intrusive load monitoring in households". In: *Scientific Data* 7. DOI: 10.1038/s41597-020-0434-6.

Ku, Wen-Yang and J. Christopher Beck (2016). "Mixed Integer Programming models for job shop scheduling: A computational analysis". In: *Computers & Operations Research* 73, pp. 165–173. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.04.006. URL: https://www.sciencedirect.com/science/article/pii/S0305054816300764.

Lai, Guokun et al. (2017). "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks". In: *CoRR* abs/1703.07015. arXiv: 1703.07015. URL: http://arxiv.org/abs/1703.07015.

Lee, Eunji and Hyokyung Bahn (2014). "A genetic algorithm based power consumption scheduling in smart grid buildings". In: *The International Conference on Information Networking 2014 (ICOIN2014)*, pp. 469–474. DOI: 10.1109/ICOIN.2014.6799726.

Maharjan, Sabita et al. (Mar. 2013). "Dependable Demand Response Management in the Smart Grid: A Stackelberg Game Approach". In: *Smart Grid, IEEE Transactions on* 4, pp. 120–132. DOI: 10.1109/TSG.2012.2223766.

– (2016). "Demand Response Management in the Smart Grid in a Large Population Regime". In: *IEEE Transactions on Smart Grid* 7.1, pp. 189–199. DOI: 10.1109/TSG.2015.2431324.

Mao, Hongzi et al. (2016). "Resource Management with Deep Reinforcement Learning". In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. HotNets '16. Atlanta, GA, USA: Association for Computing Machinery, pp. 50–56. ISBN: 9781450346610. DOI: 10.1145/3005745.3005750. URL: https://doi.org/10.1145/3005745.3005750.

Mary Mammen, Priyanka and Hareesh Kumar (Oct. 2019). "Explainable AI: Deep Reinforcement Learning Agents for Residential Demand Side Cost Savings in Smart Grids". In: URL: https://arxiv.org/pdf/1910.08719v2.pdf.

Mnih, Volodymyr et al. (Feb. 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518, pp. 529–33. DOI: 10.1038/nature14236.

Palensky, Peter and Dietmar Dietrich (Sept. 2011). "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads". In: *Industrial Informatics, IEEE Transactions on* 7, pp. 381–388. DOI: 10.1109/TII.2011.2158841.

Qayyum, Fatima et al. (Jan. 2015). "Appliance Scheduling Optimization in Smart Home Networks". In: *IEEE Access* 3, pp. 1–1. DOI: 10.1109/ACCESS.2015.2496117.

Rashid, Tabish et al. (2018). "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". In: *CoRR* abs/1803.11485. arXiv: 1803.11485. URL: http://arxiv.org/abs/1803.11485.

Sunehag, Peter et al. (2017). "Value-Decomposition Networks For Cooperative Multi-Agent Learning". In: *CoRR* abs/1706.05296. arXiv: 1706.05296. URL: http://arxiv.org/abs/1706.05296.

Zhao, Zhuang et al. (2013). "An Optimal Power Scheduling Method for Demand Response in Home Energy Management System". In: *IEEE Transactions on Smart Grid* 4.3, pp. 1391–1400. DOI: 10.1109/TSG.2013.2251018.