POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master Degree In Computer Engineering

# Currricular Internship Report


**Relatore**
prof. Chiasserini Carla Fabiana

**Correlatori:**
Somreeta Pramanik
Puligheddu Corradoi

**Candidate**
Tung Pak Man
matricola: S245432

Academic Year 2019 – 2020

# Contents

# Chapter 1

# Introduction

Hermes Reply is one of the Company parts of Reply Group specialized in innovative technological solutions, process consulting in the Manufacturing field and the maintenance services for application The company operates as a System Integrator between ERP level and Shop Floor solutions, for all management of production activities inside the factory, guaranteeing that customers achieve their business objectives. There are three main areas where The main areas where Hermes Reply operates are the following:

1. *TURN KEY PROJECTS*

IT solution's implementation dedicated to the management of production processes in the plant(Planning and Control, Execution, Quality, Manufacturing Operations Management, Shop Floor Integration, Suppliers Management) using both market-ready or customized solutions, it built according to specific customer requirements.

2. *BUSINESS CONSULTING*

Customer support in the implementation and adoption of IT solutions (As-Is analysis, Requirements identification, To-be definition), within the scope of process analysis and functional analysis activities.

3. *SUPPORT SERVICES*

Provision of specialized 1st, 2nd and 3rd Application Maintenance, in production

plants and manufacturing facilities. A consolidated management methodology, dedicated Service Maintenance tools and varying Service Agreement levels delivered 7x24, complete the team's technological and process-oriented expertise, in order to guarantee a reliable and high-quality service.

HERMES REPLY is an important partner of customers within the Automotive, OEM and discrete manufacturing industries, with a strong emphasis on "Industry 4.0" initiatives and delivery of new generation innovative IT Manufacturing solutions based on Cloud, IoT and Human Machine Interface technologies.

**Aim of the Internship**

The materials management process foresees, as part of the factory production process, the call of the materials that need to be transported from the preparation areas (kitting) to the product assembly line in order to carry out the processes necessary to obtain the finished product at end of the line. Once the logistic operators receive the input for the preparation of the materials, they place them in special trolleys that manually guide them to the destination stations that must be served. In this sense, the following path must be optimized both on the basis of the best route to take and on the basis of priorities (urgencies/needs) indicated by the assembly line. It is also necessary to ensure that the estimated time of arrival of the necessary material is available on the destination stations, through special operator panels. To create a solution that learns and offers operators the best paths to take on the basis of experience and what is happening in the surrounding environment, the adoption of machine learning technologies that provide reinforcement learning algorithms will be studied.

The ultimate goal is to obtain experimentation of the technology and a proof of concept that demonstrates the possibility of adopting this solution in the logistics of the production plant.The main activities in which this Internship will be involved are related to:

- Introduction to the customized software (PMS suite) which designed for complete manufacturing coverage.

- The implementation of the experimentation which consists of 'phases of analysis', 'choice of algorithm', 'training' and 'testing'.

### Content of the Report

- Chapter 2 gives an overview of the PMS suite developed by the company for monitoring and controlling the production plants.
- Chapter 3 describes the scenario and related machine learning technologies. There is a brief introduction to the creation of the machine learning model.
- Chapter 4 analyses the possibility of further exploring the thesis work by applying reinforcement learning algorithms on the network condition.

# Chapter 2

# PMS Suite

In today's competitive mart, manufacturers need to boost efficiency to gain an advantage. In order words, they need to '' make more with less', this is what the PMS suite can do. It is a collection of different systems designed for complete manufacturing coverage. In technical terms, they are systems that connect and monitor machines and work centers on the factory floor. It enables an efficient combination of production and logistics. This results in a high level of customer satisfaction and releases tied-up capital for investments. Production machines can be linked to the PMS suite , thus Material transports are synchronized with the production processes. In this way, demand-driven material supply to machines and markets with the precise parts needed at a given point in time, and goods collection, through cranes, forklifts, tugger trains, conveyor systems, and so on can be secured. The main goal is to ensure effective execution of manufacturing operations and increase production efficiency. The PMS suite is composed of three systems:

- *Production Management System*
  It is a Manufacturing Execution System(MES) for Engines and Transmissions plants. MES can be linked to different external SFS systems for controlling and monitoring purposes.

- *Operator Terminal* It is a Shop Floor System (SFS) that guides line operators in their work. It receives and sends messages to MES.

- *CALL* It is a material management module of PMS suite for lines supply.

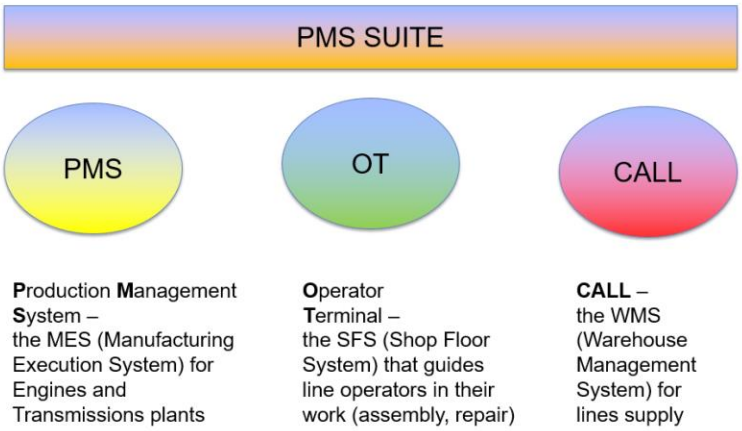



Figure 2.1: .
PMS suite

The Machine-Learning module will be applied in the process context of CALL system .

An example can be a rocket launching event, the MES is like the mission control center. The monitor can track the power, temperature and other status about the rocket. The SFS is like a gate controller. When the MES authorizes the launch, it will communicate with SFS by means of a message to open the gate. The CALL is responsible for resource supply, e.g to recharge the fuel.

**Manufacturing Execution System(MES)**

Manufacturing Execution Systems (MES) are software that ensures quality and efficiency are built into the manufacturing process and are proactively and systematically enforced. It connect multiple plants ( or signle plant), sites and vendors' live production information, and integrate easily with equipment, controllers and company business applications. Resulting in a complete visibility, control and manufacturing optimization of production and processes across the company.

It offers a lot of functionality. The workers can have a global view of planned production schedules and their production routines. This ensures all the workers are on the same page and reduces errors due to misunderstanding. Grouping final parts with all their corresponding manufacturing data such as source provider and id number— from the raw material in ‘’maching” area to the ‘finished product in ‘’assembly”’area. This data is especially useful for manufacturers that must comply

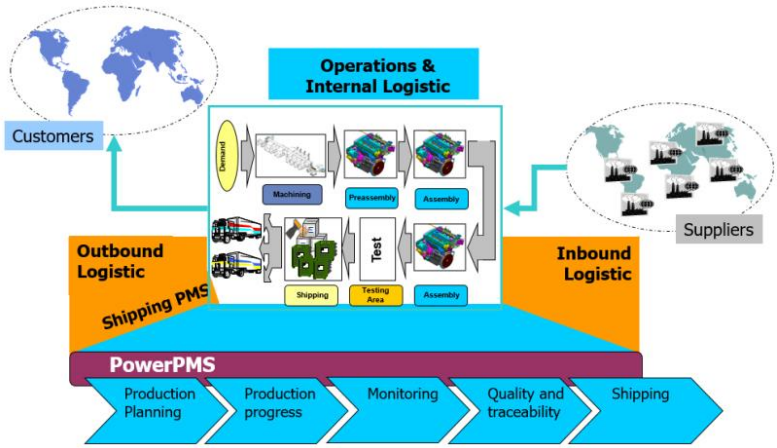



Figure 2.2: .
PMS suite

with government or industry regulations. Managing the quality of the manufacturing process and units including quality deviations and exceptions will be easier than ever. More easily and accurately plan machine maintenance to reduce downtime and production interruptions. Tracking and gathering essential data and easily recall that data when needed.

MES provides a number of long and short-term benefits.

- Eliminate non-value-added activities
- Standardize processes across all sites
- Gain real-time visibility and control across the plant
- Lower cost of good quality
- Make fact-based operational and strategic decisions easily
- Complete control over the manufacturing process

### Shop Floor System (SFS)

SFS is a software system of methods and tools that are used to schedule, track, and report on work in a manufacturing plant. It generally evaluates the portion of operation in a line that are completed. These status are useful for inventory evaluations, resource planning, and supervisor and operator productivity on a shop floor.

Data reported and gathered by SFS regarding labor, production time and work progress can also improve worker productivity, reducing cycle times and throughput. In addition to provide worker productivity information to monitors, real-time SFS can be helpful to the operators themselves. Giving them access to their own work process improves the ability for them to set, meet and exceed their goals. In summary, SFS within MES improves the productivity of any shop,in all manufacturing styles or capacity.

# Chapter 3

# Background Knowledge

To maintain good productivity in a plant, the line should not stop for any reason. A stop in the line could cause the company to lose a lot of money. To ensure the line is working, the first thing is to make sure that there is a continuous supply for the material used in the line. The first difficulty is how to move the material from the stock area to the line. There are many considerations and limits when designing the solution. For example, time is the most important aspect that if the material is not delivered in time, the production line will stop.Besides, the solution should be generalized, which should work independently without considering the physical area of the plant as the map area will change easily.The idea of the project is to provide a logistic support tool for those collection and deliverance processes within the plant. The aim is to optimize those processes in the collection and delivery phases in terms of time and resources, provide forecasts and monitor the process in order to avoid mistakes.

**Objective**

There are three main targets in the tool.

- *Means of transport*
  It should be able to command several operators at the same time and choose the best combination of operators to send based on the capacity for each order. It should also minimize the number of traveling operators and Take into account the different types of operators available at the moment.

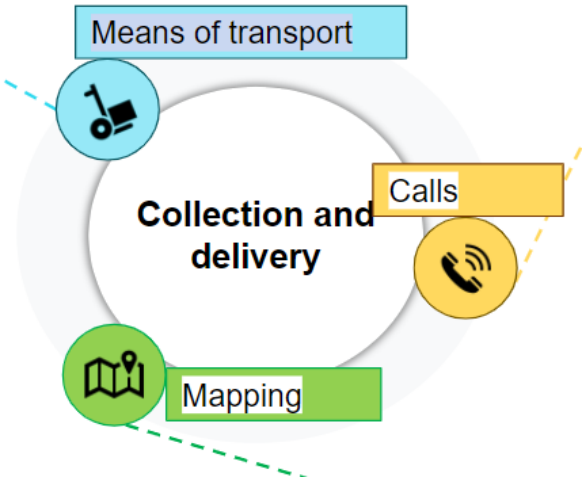



Figure 3.1: .
The above three aspects are the main areas the tool focusing on.

- *Map*
  A mission should be able to start from anywhere and arrive anywhere in the plant. It should avoid temporarily blocked paths due to errors or technical problems. The model should take into account privileged directions and one-way path.

- *Call*
  The mission should be able to start within a fixed time interval or after a certain number of calls made. Priority labels can be added to any mission. The tool should provide in advance and in real-time a temporal forecast of when a mission will be completed. It should also monitor the path of operators and check deliveries have been correctly carried out.

Based on the calls it has in charge, the system proceeds to find the best combination of operators available to manage in collection phases and carry out missions in the shortest possible time and using the fewest operators. The calculation of the best combination can take place: after a certain time, at each call, or after a defined number of calls. These methods can be mixed to determine the best combination for the establishment. The system proceeds to provide an estimate of

the expected timescales for completing the mission, the monitoring of operators, and checking the consistency of delivery. It is possible at any time to raise the priority of a delivery so that it is immediately managed. It takes into account the maximum time for which a station can wait. The system can also modify the routes of ongoing missions based on what is happening, always in compliance of the balances of the process. In the event that there are planned or unforeseen changes in the plant that make certain routes unusable, the system should avoid such obstacles.

Resulting benefits are :

- *Optimization*
  Optimization of travel time and path length. Reduction of the number of operators used. The possibility to request the delivery priority.
- *Forecast*
  Forecast of withdrawal and delivery time.
- *Monitoring*
  Monitoring of positions and actions of operators and checking the correct completion of missions in the intended destinations.

**Proof of concept**

- Creation of a virtual map of a generic plant, with several points ofcollection and delivery.
- Definition of a number of operators with different skills.
- Creating calls with different collection and delivery points . The system must able to find the best combination of routes and operators, providing an estimate oftimes of the missions
- Insertion of obstacles in the path and demonstration of how the system proceeds to avoid them.

### Reinforcement Learning

The tool is Reinforcement Learning, a particular Machine Learning technique developed to help the machine learn the environment in which it works. It is an ML technique applied to an agent who, by receiving stimuli from the outside, operates in an environment and needs a series of moves to achieve a goal.

- *What does it do ?*
  Like most Machine Learning-based processes, it identifies the best logical paths and most important features of our data in order to make the best decisions. The peculiarity of the RL is given by the interaction with the environment: it receives input and the current environment and returns answers.

- *When is it useful?*
  When you have a large number of very complex data that a classical algorithm could not calculate.
  When there are variables that are difficult to predict ( problems with machinery, human errors, etc ...).
  When a dynamic system is required.

- *How does it work?*
  The algorithm is developed through a series of simulations in an environment, often reproduced virtually, with a large number of possible variations Rewards and penalties are awarded for actions performed by the agent while interacting with the environment. The 'policy' is processed by classical Machine Learning algorithms, mainly by a neural network.

- *Markov Property*
  A certain state information contains all relevant history, as long as the current state is known, all historical information is no longer needed, and the current state can determine the future, the state is considered to be Markov.

$$P[S_{t+1|S_t}] = P[S_{t+1}|S_1, ....., S_t]$$

- *Markov Process*
  It is a random process without memory, which can be represented by a tuple

<S,P>, where S is a finite number of state sets, and P is a state transition probability matrix.

- *Markov Reward Process*
  Markov reward process adds reward R and attenuation coefficient y : <S,P,R, y > on the basis of Markov process. R is a reward function. The reward in the S state is the reward expectation that can be obtained at the next time (t+1) in the state s at a certain time (t)

$$R_s = E[R_{t+1}|S_t = s]$$

- *Return*
  Return is the decayed sum of all rewards from time t on a Markov reward chain.

$$G_t = R_{t+1} + y * R_{t+2} + y^2 * R_{t+3} + .....$$

- *State-Value Function*
  The state value function gives the long-term value of a certain state or action. State Value Function can only describe the state, it can also describe a certain action in a certain state, and in some special cases, it can also describe only a certain action.

$$v(s) = E[G_t|S_t = s]$$

- *Markov Decision Process*
  Compared with the Markov reward process, the Markov decision process has one more Action set A, which is a tuple like this: <S, A, P, R, y>. It looks very similar to the Markov reward process, but here P and R correspond to the specific action a, instead of only corresponding to a certain state like the Markov reward process. A represents a limited set of actions.

$$P^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a]$$

$$R^a_s = E[R_{t+1}|S_t = s, A_t = a]$$

- *Policy*
  Policy $\pi$ is a set or distribution of probabilities, and it is the probability of taking action a a for a certain state s in the process.A Policy completely defines the agent's action, that is to say, defines the agent's various possible actions in each state and the magnitude of its probability. Policy is only related to the current state, and has nothing to do with historical information; at the same time, a certain policy is static and has nothing to do with time; but the agent can update the policy over time.

$$\pi(a|s) = P[A_t = a|S_t = s]$$

  **If the policy is given by another machine learning model, such as a neural network, the parameter of the policy is the weight coefficient.**

- *State-Value Function based on Policy*
  It is the state value function based on the policy $\pi$ under the MDP, which represents the expectation of gains obtained when starting from the state s and following the current policy; in other words, when the current policy $\pi$ is executed, the return value when the agent is in state s.Policy is static and concept of the whole, which does not change with the change of the state; the change is the specific action that may occur according to the strategy in a certain state, because the specific action has a certain probability, and the policy is used to describe each The probability of performing different actions in different states.

$$v_\pi(s) = E[G_t|S_t = s]$$

- *Action-Value Function based on Policy*
  It represents the expectation of the gains that can be achieved by performing a specific action a on the current state s when the policy $\pi$ is implemented; or when following the current policy $\pi$, it measures the value of performing action a on the current state. The behavior value function generally corresponds to a specific state, and a more detailed description is the state & action versus the value function.

$$q_\pi(s, a) = E[G_t|S_t = s, A_t = a]$$

- *Optimal state value function*
  The optimal state value function refers to the selection of the function that maximizes the value of state s among the state value functions generated from all policy

$$v_* = max_\pi v_\pi(s)$$

  Similarly, the optimal action-value function refers to the selection of the function that maximizes <s,a> among all the action-value function generated from all policy.

$$q_*(s,a) = max_\pi q_\pi(s,a)$$

  The optimal state value function clarifies the best possible performance of the MDP. When the optimal state value function is known, the optimal value of each state is also known. At this time, the MDP is assumed to have been solved.

- *The best policy*
  When for any state s, the value of following policy $\pi$ is not less than the value of following policy $\pi$', then policy $\pi$ is better than policy $\pi$' For any MDP, the following points are true: 1. There is an optimal policy that is better or at least equal to any other policy; 2. All optimal policy have the same optimal state value function; 3. All optimal policy have the same action value function.

# Chapter 4

# Experiment

The main target is divided into a few small targets .

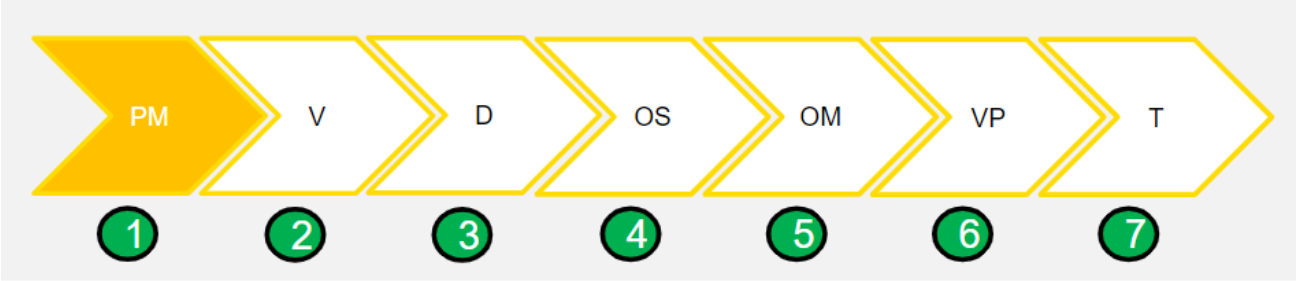



Figure 4.1: .
The final solution is constructed step by step starting from an easier target.

- *PM (percorso migliore)*
  Mouse in the maze.Finding the best path within a labyrinth (outer edges and inner limitations) given a starting point and destination to an element.

- *V (vincoli)*
  Adding direction constraints

- *D (destinazioni)*
  Adding more destination points for the single mouse in the labyrinth: it must arrive in all points in the shortest time possible.

- *OS (operatore-singola destinazione)*
  Adding more operators with the same goal (point of destination) by starting them one after the each other

- *OM (operatore-destinazioni multiple)*
  Adding more operators with different target points

- *VP (variabilità di processo)*
  Addition of variability of process (priority, someone stops)

- *T (tempo)*
  Added estimated time of arrival(countdown)

- Contextualization in the customer process (the labyrinth becomes the map of the plant) . Then programming of events that can trigger the mission generation (e.g. piece ready)

### Introduction on the Algorithm

- Initialize the map , 1 is free and 0 is the wall.

```
In [2]: maze = np.array([
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
            [ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  0.,  1.,  1.],
        ])
```

Figure 4.2: .

- Define possible actions

```
In [3]: # Gray scale marks for cells
        visited_mark = 0.9
        flag_mark = 0.65
        agent_mark = 0.5

        # Actions dictionary
        actions_dict = {
            0: 'left',
            1: 'up',
            2: 'right',
            3: 'down',
        }

        num_actions = len(actions_dict)
```

Figure 4.3: .

- Create a training activity. Randomly choose a starting point and endpoint. Keep 'playing' the game if it is not game over.

```
def train(self):
    start_time = datetime.datetime.now()
    self.seconds = 0
    self.win_count = 0
    for epoch in range(self.n_epoch):
        self.epoch = epoch
        self.loss = 0.0
        agent = random.choice(self.agent_cells)
        print("Random agent cell: {}".format(agent))
        self.env.reset(agent)
        game_over = False
        # get initial env_state (1d flattened canvas)
        self.env_state = self.env.observe()
        self.n_episodes = 0
        while not game_over:
            game_over = self.play()

        dt = datetime.datetime.now() - start_time
        self.seconds = dt.total_seconds()
        t = format_time(self.seconds)
        print("Flags set: {}".format(env._flags))
        print("Target position: {}".format(env.target))
        fmt = "Epoch: {:3d}/{:d} | Loss: {:.4f} | Episodes: {:4d} | Wins: {:2d} | flags: {:d} | e: {:.3f} | time: {}"
        print(fmt.format(epoch, self.n_epoch-1, self.loss, self.n_episodes, self.win_count,
                         len(self.env.flags), self.epsilon(), t))

        if self.win_count > 2:
            if self.completion_check():
                print("\n")
                print("Completed training at epoch: %d" % (epoch,))
                break
            else:
                print("\n")
        else:
```

Figure 4.4: .

- Define the play activity.To play a game, randomly choose an action. The Markov Decision Process is then called to evaluate the state.

```
def play(self):
    action = self.action()
    prev_env_state = self.env_state
    self.env_state, reward, game_status = self.env.act(action)
    if game_status == 'win':
        self.win_count += 1
        game_over = True
    elif game_status == 'lose':
        game_over = True
    else:
        game_over = False

    # Store episode (experience)
    episode = [prev_env_state, action, reward, self.env_state, game_over]
    #print(memory_ep)
    self.experience.remember(episode)
    self.n_episodes += 1

    # Train model
    inputs, targets = self.experience.get_data(data_size=self.data_size)
    epochs = int(self.env.base)
    h = self.model.fit(
        inputs,
        targets,
        epochs = epochs,
        batch_size=16,
        verbose=0,
    )
    self.loss = self.model.evaluate(inputs, targets, verbose=0)
    return game_over
```

Figure 4.5: .

- Define Markov Decision Process

```
def get_data(self, data_size=10):
    env_size = self.memory[0][0].shape[1]   # env_state 1d size (1st element of episode)
    mem_size = len(self.memory)
    data_size = min(mem_size, data_size)
    inputs = np.zeros((data_size, env_size))
    targets = np.zeros((data_size, self.num_actions))
    for i, j in enumerate(np.random.choice(range(mem_size), data_size, replace=False)):
        env_state, action, reward, next_env_state, game_over = self.memory[j]
        inputs[i] = env_state
        # There should be no target values for actions not taken.
        # Thou shalt not correct actions not taken #deep (quote by Eder Santana)
        targets[i] = self.predict(env_state)
        # Q_sa = derived policy = max quality env/action = max_a' Q(s', a')
        Q_sa = np.max(self.predict(next_env_state))
        if game_over:
            targets[i, action] = reward
        else:
            # reward + gamma * max_a' Q(s', a')
            targets[i, action] = reward + self.discount * Q_sa
    return inputs, targets
```

Figure 4.6: .

- Policy refine. A sequential neural network is chosen to refine the policy. The output is the set of moves and their weights (used in the policy) in the actual position.

```
n [7]: def build_model(env, **opt):
           loss = opt.get('loss', 'mse')
           a = opt.get('alpha', 0.24)
           model = Sequential()
           esize = env.maze.size
           model.add(Dense(esize, input_shape=(esize,)))
           model.add(LeakyReLU(alpha=a))
           model.add(Dense(esize))
           model.add(LeakyReLU(alpha=a))
           model.add(Dense(num_actions))
           model.compile(optimizer='adam', loss='mse')
           return model
```

Figure 4.7: .

- The training process for the actual map is divided into a few sub-maps to avoid huge training time (weeks). Then each sub solution is combined by simple codes.

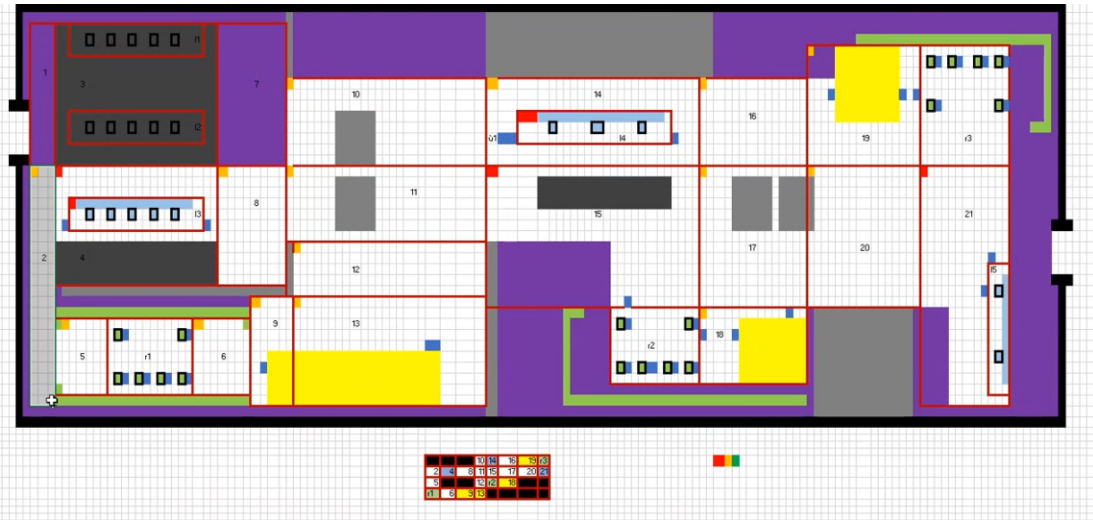
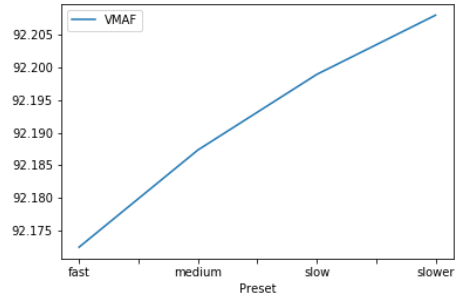


Figure 4.8: .

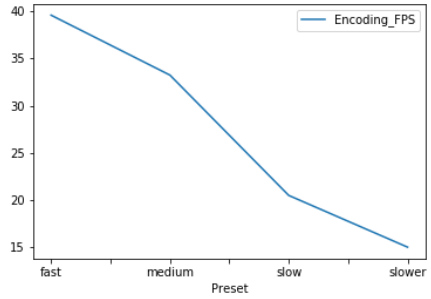
**Conclusion**

At the end of the completion of the experimentation, we will have a web page to demonstrate how the trained model will be applied to choose the better operator and the better path depending on the static variables but also on the dynamic variables like the positions of the operators, needs of the lines and all the others related to the environment.

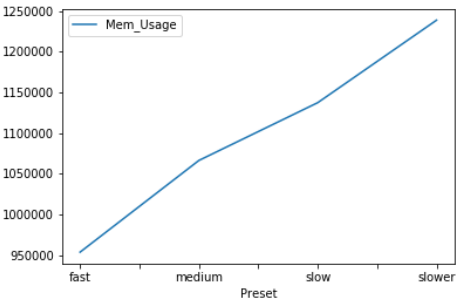
**Further exploring on the thesis work with Reinforcement Learning**



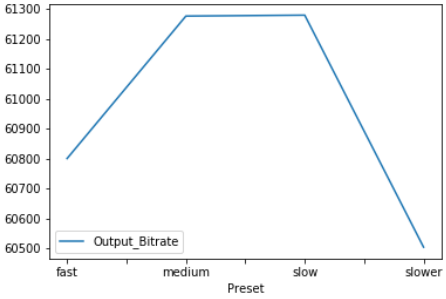

(a) Preset - Vmaf




(b) Preset - Encoding FPS




(c) Preset - Mem Usage




(d) Preset - Output bitrate

Figure 4.9

Recall from Section 5.3 Data Analysis from the Thesis , The variation in preset value induces only a neglectable impact on VMAF score. Increase the preset value can increase encoding speed and reduce memory usage significantly. In theory, the file size will become larger is a faster preset is used. If the network condition is ignored, the preset value can always set to'veryfast' in order to speed up the encoding process. In practical conditions, the network condition can not be ignored. If the network is idle, the preset can always set to 'veryfast'. However, if the network

is saturated, attention should be needed on choosing the preset value to avoid the potential of having a bottleneck during transmission. Besides that, the priority label can be also added to the stream based on the client type, some clients may want to become a VIP if they want a faster connection or higher quality. Reinforcement Learning is designed for dealing with those dynamic environments. The model can train on a simulated environment first, then we can deploy it to the real environment and let it improve on the fly.

# Acknowledgements

Foremost, I wish to express my deepest gratitude to Professor CHIASSERINI CARLA FABIANA for giving me a chance in doing an Internship. Her patient guidance, enthusiastic encouragement and useful critiques of this thesis, always steering me in the right direction.

Secondly, I would like to thank Mr. Casetta Andrea and Mr. Franco Catanzariti. I appreciated the chance you give me to learn and grow in the company. Their availability in answering and communicating even during the holiday, without this, it's hard to accomplish the mission.

Last but not the least, a special thanks to everyone who has supported and encouraged me on this journey.