A Thesis

For the Degree of Doctor of Philosophy

Efficient Task Management Mechanism Based on Learning to Scheduling in Smart Factory

Sehrish Malik

Department of Computer Engineering

Graduate School

Jeju National University

December 2019

Efficient Task Management Mechanism Based on Learning to Scheduling in Smart Factory

Sehrish Malik
(Supervised by Professor Do-Hyeun Kim)

A thesis submitted in partial fulfillment of the requirement for the degree of Doctor of Philosophy in Computer Engineering

2019. 12. 18

This thesis has been examined and approved.

안 기 중 [illegible]
....
Thesis Director
Khi-Jung Ahn, Professor, Department of Computer Engineering

송 왕 철
....
Wang-Cheol Song, Professor, Department of Computer Engineering

[illegible]
....
Yun-Jung Lee, Professor, Computer Science and Statistics

[illegible]
....
Jung-won Cho, Professor, Department of Computer Education

[illegible]
....
Thesis Supervisor
Do-Hyeun Kim, Professor, Department of Computer Engineering

December 18th, 2019

Department of Computer Engineering

GRADUATE SCHOOL

JEJU NATIONAL UNIVERSITY

Acknowledgement

This study is wholeheartedly dedicated to my beloved father Malik Abid Hussain and my mother Farda Abid; they both have always been a true source of inspiration for me. I would like to mention the role of my close friend Wafa Shafqat, for becoming my family abroad and being with me through thick and thin. I wish to thank my siblings Malik Faisal Sohail, Malik Nouman Abid, Aneeqa Malik and Saira Hussain, for always supporting me in my decisions.

I wish to express my sincere appreciation to my supervisor, Professor Do-Hyeun Kim, who convincingly encouraged me to be professional in achieving my goals and directed me to do the right thing even during my low times. Without his persistent help, the goal of completing this thesis wouldn't have been possible. I would like to acknowledge that he played a vital role in shaping my views towards maintaining a healthy balance between professional and personal life goals.

I wish to appreciate the support of my colleagues, during my study at Jeju National University. I am grateful to my seniors Dr. Israr Ullah, Dr. Muhammad Fayyaz and Dr. Wenquan Jin for their guidance at early stages of my research. I would like to extend my gratitude to my lab mates Shabir Ahmad, Faisal Mehmood, Hang-Lei, Faisal Jamil, Imran Jamal, Azimbek, Young-Uk and Naeem Iqbal for providing a very professional and friendly working environment.

I greatly appreciate the assistance and exposure provided by Jeju National University and its Computer Engineering Department in fulfilling my dreams.

Dedicated to

My Father, Malik Abid Hussain ♥

(16/04/1962 – 01/02/2019)

Table of Contents

List of Figures

List of Tables

Abstract

Smart factory also known as smart manufacturing is an emerging field with the revolution of industry 4.0. The smart factory concept is an integration of internet of things technologies, computing platforms, cyber-physical systems, control mechanisms, data modeling and simulations, optimization techniques and predictive engineering. With the help of all these concepts, the smart factory integrates the manufacturing assets and represents industrial networks. The aim of smart factory industrial networks is mass customization, on-demand supply chain management, optimal and flexible processing solutions, and parallel processing. Smart factory faces many limitations in the current age and is need of research solutions for issues such as environmental hazards, energy consumption, productivity, efficient planning, task management, job scheduling, machine utilization, reliable infrastructure and integrated solutions.

In this thesis, we put our efforts to find integrated solutions for smart factory concerns by proposing an efficient task management mechanism based on learning to scheduling in smart factory. The scope of the proposal is to efficiently plan tasks execution, maximize machines' resource utilization, maximize productivity, minimize production delays, efficiently handle exceptions and efficiently control smart factory actuators. The proposed learning to scheduling mechanism focuses on both machine structure and tasks modeling for efficient scheduling. We design and develop an integrated solution of learning to scheduling based on sub-modules of prediction and learning for prediction mechanism, optimization and learning for optimization mechanism and inference engine based control mechanism in this thesis work.

The scheduling algorithm used for the efficient task management is hybrid of the two scheduling approaches as agent cooperation mechanism (ACM) and fair emergency first (FEF) scheduling scheme. ACM is a decentralized scheduling approach which focuses on the

production maximization goals per machine and also centers the production goals of all the machine networks involved in the smart factory. FEF scheduling scheme focuses on minimizing the tasks starvation rate and maximizing the machine utilization by efficiently using the machine slots. In FEF scheduling scheme, two predictive learning based factors are used to improve the scheduling performance; UM (Urgency Measure) and FM (Failure Measure). Both UM and FM use ANN prediction algorithm to learn from scheduler's history decisions and put the learnings in context to wisely use the free machine slots; aiming to increase machine utilization without risking timely execution of any high priority task.

The learning to prediction mechanism takes scheduler history data as input and predicts the future tasks completion status and machine utilization rate under varying tasks' loads. The prediction algorithm used is artificial neural network (ANN) and learning algorithm used is particle swarm optimization (PSO). The learning algorithm of PSO tunes the ANN's weights during training iterations to optimize the ANN weights and maximize the prediction accuracy.

The learning to optimization mechanism aims to maximize the machine utilization for machines involved in the smart factory, in order to efficiently use the machine resources. The optimization algorithm used is PSO and the learning algorithm used is ANN. First, PSO history is built to train the ANN algorithm and then based on ANN training the PSO particle's velocities are tuned in order to enhance the optimization results.

The control mechanism for smart factory actuators is based on the inference engine. The inference engine is fed with rule base which contains list of rules for existing actuators based on incoming sensing and system values. The inference engine matches the rules and generates the control tasks. The control tasks are sent to the scheduler to be executed; and on execution of control tasks, the control commands are sent to the actuators via control unit.

The proposed task management mechanism is evaluated based on multiple scenario simulations and performance analysis. The comparisons analysis shows that proposed task management system, referred as learned predictive and optimized hybrid scheduling scheme, significantly improves the machine utilization rate and drastically drops the tasks instances missing rate and tasks starvation rate. Overall, we observe that the learned predictive FEF scheduling in comparison to basic FEF scheduling scheme shows an average of 72.23% reduction in tasks starvation rate and an average of 54.17% reduction in tasks instances missing rate reduction. Also, the learned predictive and optimized hybrid scheduling scheme demonstrates an average of 27.28% increase in machine utilization, and an average of 36.38% improvement in response times.

Chapter 1: Introduction

Technology has vastly changed with the changes witnessed by the industrial revolutions. Till today, the world has observed four industry revolutions where first industrial revolution started in 1784 with the introduction of mechanization based on steam and water power. The second industrial revolution came in 1923 with the introduction of mass production and electricity. The third industrial revolution came in 1969 with the introduction of electronics and IT systems along with introduction of automation. The fourth industrial revolution came in 2014 with the introduction of cyber physical system based on smart machines, sensors, automated control of actuators, and inter-connectivity between the physical world the virtual world [1]. The 4th industrial revolution resulted in the introduction of smart manufacturing and smart factory. The Industry 4.0 can be defined as smart factories with connected machine and intelligent robots based on the cyber physical systems.

The underlying technology of smart factory is internet of things (IoT) which has smart sensing technologies, smart machines connected to network, intelligent and automated control with self-awareness, self-prediction, self-optimization, self-configuration and self-diagnosis. Such systems are enabled with the help of cyber physical system (CPS) concept. The integration of IoT and CPS into factory creates a virtual twin of the physical world with each physical object having its virtual representation. In a smart factory all the objects are connected using the IoT networks and the operations are operated by CPS. The basic goal of IoT is to connect the real-world objects while the aim of CPS is to connect the physical world with the virtual world [3] as shown in Figure 1.

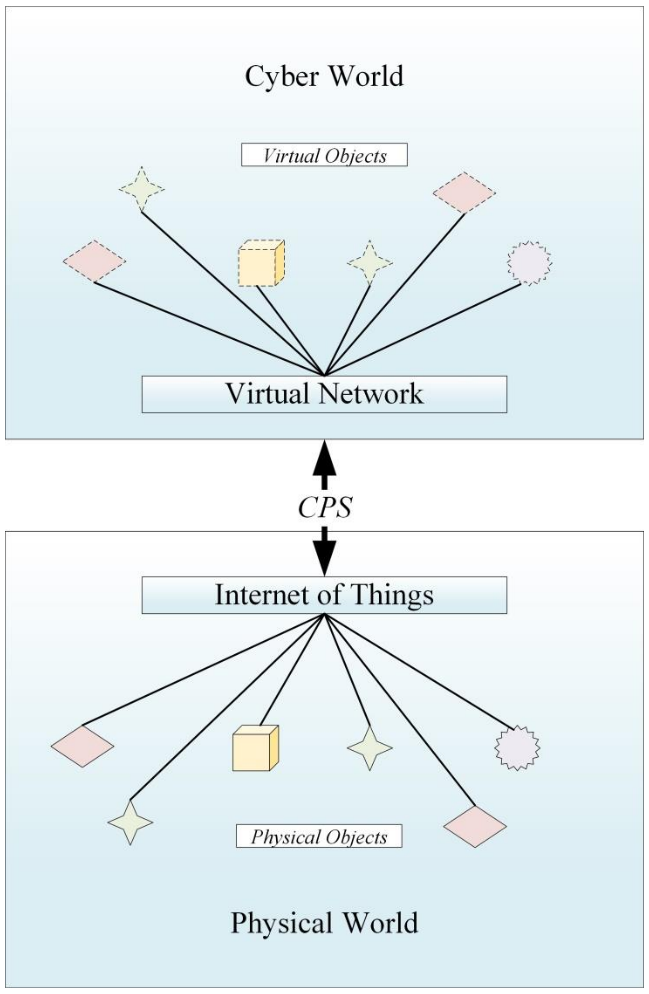


Figure 1: Integration of IoT and CPS

The virtual objects are contained in a virtual network which replicates the physical representation, dependencies and context of the physical world objects. The IoT enabled smart factory solutions help achieving the real-time production visualization with the identification of manufacturing objects. The technologies such as radio frequency identification (RFID) are used to interpret the real-world object into smart factory's virtual objects along with their behaviors and interactions. The development of such system facilitates in the smart factory production process, intelligent decision making and automated control process, and other operations [4].

The smart machines participate in generating huge volumes of data known as big data. Big data can be used and analyzed to aid the smart factory production. Artificial intelligence (AI) and machine learning mechanisms are used to interpret the big data into useful information to be applicable. The right use of big data can help smart factory to optimize the production by maximizing the production output, maximizing the machine utilization, minimizing the energy consumption, minimizing the production cost, and minimizing the production time. The application of AI and machine learning into big data can result into application scenarios such as predictive maintenance, fault detection, product's quality detection, production cost predictions etc. The big data analytics are deployed for performance monitoring, performance management and optimization of the operations.

The smart factory has inter-connected supply chains and autonomous control of vehicles, machines and robots resulting in efficient production tasks management such as getting shipments ready based on tracking of ships' arrivals and departures and avoiding delays with the help of self-driving vehicles and self-delivering robots. The goal of smart factory is to deliver smart solutions to the customers of smart factory.

Customers play a vital role in the smart factory. In order to assure the customers' satisfaction and growth, it very crucial for the smart factory to perform the production process efficiently and effectively in real-time by meeting all constraints [5]. This task can be performed with the help of following two factors set at the right place. The first is the automated feedback of the production processes and second is the implementation of analytics tools to accurately predict the production and consumers patterns [6]. Hence, the predictive and optimized scheduling is very crucial for the smart factory's timely task management.

In this thesis, we propose learning to scheduling mechanism based on learning to prediction and learning to optimization in smart factory.

The proposal aims to aid the smart factory's manufacturing processes with efficient tasks allocation, efficient tasks dispatching and efficient tasks scheduling; in order to improve the overall productivity of the manufacturing process. The proposed mechanism involves the scheduling of tasks with the aid of learning modules of prediction and optimization. The learning modules provide the history based predictions and tuned parameters for improved task management. In the Figure 2 show, we present the conceptual diagram of the proposed system.

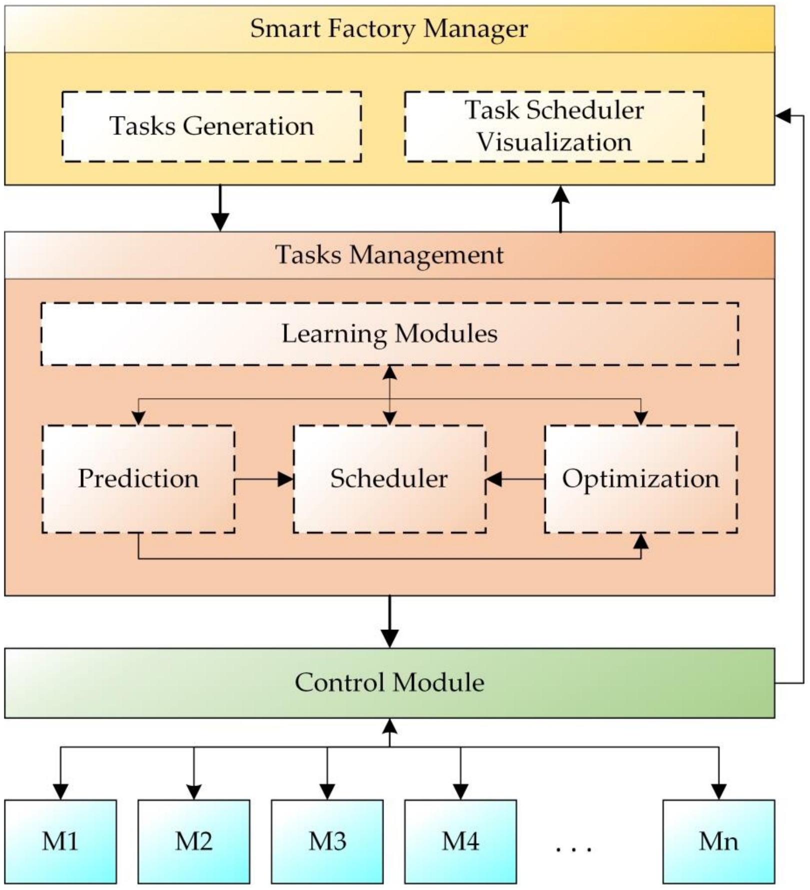


Figure 2: Conceptual diagram of the proposed task management mechanism

The proposed solution is based on three main modules as prediction module, optimization module and scheduling module. The prediction and optimization module facilitate the scheduling module by improving the task management process with their learned inputs. The scheduling scheme for efficient task management is based on the hybrid decentralized scheduling named as agent cooperation mechanism (ACM) and a fair emergency first (FEF) scheduling scheme. Two learning based sub-modules are added in scheduling to improve the scheduling outcome: UM (Urgency Measure) and FM (Failure Measure). We propose an optimized prediction scheme for predicting the tasks execution status and machine utilization rate under given load of the tasks based on the history decisions. We also proposed improved variations of PSO for optimized predictions using ANN as VB-PSO-NN and R-PSO-NN. An objective function is proposed for enhancing machine utilization and to seek the optimal results based on PSO algorithm. Also, we use the proposed improved variations of PSO (VB-PSO and R-PSO) as optimization algorithm. We further implement the ANN learning based VB-PSO and R-PSO; where ANN is used to tune the particle weights.

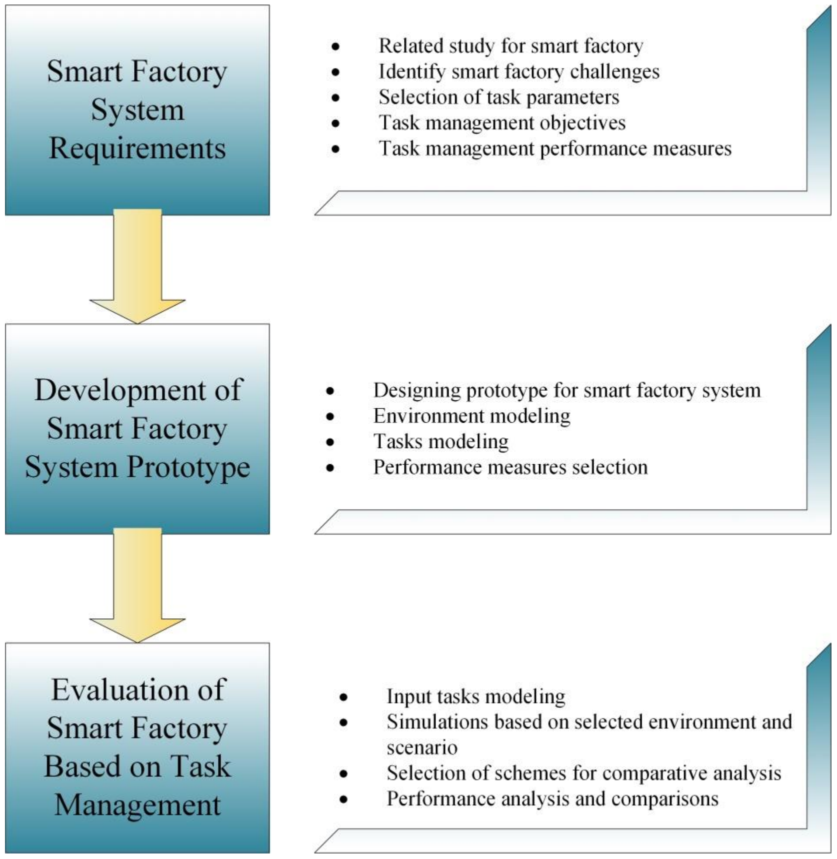

Smart Factory System Requirements
• Related study for smart factory
• Identify smart factory challenges
• Selection of task parameters
• Task management objectives
• Task management performance measures
Development of Smart Factory System Prototype
• Designing prototype for smart factory system
• Environment modeling
• Tasks modeling
• Performance measures selection
Evaluation of Smart Factory Based on Task Management
• Input tasks modeling
• Simulations based on selected environment and scenario
• Selection of schemes for comparative analysis
• Performance analysis and comparisons

Figure 3: Smart factory development phases

The development phases of the thesis study are shown in Figure 3. There are three main development phases as related study for smart factory system requirements, development of smart factory system prototype and evaluation of the smart factory based on task management.

The rest of the thesis is structured as follows. In chapter 2, we present the literature review divided into subsections as internet of things and cyber-physical system in smart factory (section 2.1), scheduling mechanisms (section 2.2), prediction mechanisms (section 2.3) and optimization mechanisms (section 2.4). In section 2.5 we present the limitations of existing works in smart factory and we also provide comparisons of existing works with proposed work. In section 2.6 we present the basics of two main algorithms used in the proposed work. In chapter 3, we present the proposed learning to scheduling mechanism for efficient task management. The section is divided into five sub-sections as conceptual learning to scheduling mechanism in section 3.1, learning to scheduling modules presented in section 3.2, learning to prediction modules presented in section 3.3, learning to optimization modules presented in section 3.4 and control mechanism presented is section 3.5. In chapter 4, we present the simulation developments environment. Chapter 4 is divided into four sub-section as environment modeling presented in section 4.1, input tasks notations presented in section 4.2, simulation implementation environment presented in section 4.3 and scheduling simulation application in presented in section 4.4. In Chapter 5, we present the simulation and performance analysis of the proposed learning to scheduling mechanism. We use task modeling scenarios as candy box factory tasks dataset, user input based simulated tasks dataset and machine cluster tasks dataset. We analyses the results based on the (a) analysis of the prediction module (b) analysis of the optimization module and (c) analysis of the hybrid scheduling scheme. The performance analysis metrics considered are prediction accuracy, tasks instances missing rate, tasks starvation rate, machine utilization rate and machine response time. In chapter 6, we conclude the thesis.

Chapter 2: Related Work

In this section, we present the related literature of smart factory. Smart factory is based on many key components such as IoT, cyber-physical system, process scheduling, prediction mechanisms for improving the outcomes based on learnings from history actions and responses, optimization mechanism for improving the outcomes based on finding the optimal parameters for certain scenarios and optimal solutions to scenario based problems.

Many changes in factory happened after emergence of industry 4.0, in form of production, supply of products and timeline [2]. The differences in industry before fourth industrial revolution and after fourth industrial revolution can be marked as changes in concepts of mass production to mass customization, scheduled supply to on-demand supply, static attributes to optimal and flexible attributes, and focus from product to usage.

The rest of the sections are divided as following. First, in section 2.1 we explain the evolution of IoT and revolution in industry with introduction of IoT, resulting in Industry 4.0 which is also be referred as smart factory. Also, in this section, we present the cyber-physical systems as these are one of the key components of smart factory. In section 2.2, we present the literature review of scheduling mechanisms in smart factory. In section 2.3, we present the prediction mechanisms used in the smart factory domain and the most commonly used prediction algorithms. In section 2.4, we present the related works to optimization mechanisms used in the smart factory domain and the most commonly used optimization algorithms. In section 2.5, we present the limitations of existing solutions and in section 2.6 we highlight the existing and related algorithms which we will be using in our proposed mechanism.

2.1 Internet of Things and Cyber Physical Systems in Smart Factory

Internet of Things (IoT) is one of the key essences of smart factory. IoT can be simply defined as integration of internet into things or objects. The IoT networks consist of embedded devices such as sensors and actuators which are connected to form sensor networks and actuator networks. The integration of IoT into factory enables the smart manufacturing process. Smart manufacturing has real-time data sharing and interaction among the smart devices, machine and objects. [11,12]. IoT elements which are widely used in smart factory include radio frequency identification technology, smart tags, sensing technologies, location tracking, real-time actuators' control etc. [13].

Smart factory uses the combination of IoT technologies and industry technologies consisting of sensors, actuators, network connectivity, smart computing, predictive analytics and optimized control [14].

The digital revolution for smart factories started after 1970s and continues till today, with rapid increase in automation and smart control of industry manufacturing by integrating first IT (Information Technology) and now IoT based technologies, models, frameworks and solutions. The complete integration of IoT technologies in factory environment is termed as the fourth industry revolution [15-18].

The inclusion of IoT technologies in factory results in boosting the factory production and efficiency. In 2014, a survey conducted by American society for quality presents that the inclusion of smart manufacturing significantly affects the factory efficiency. The results have shown the customers to be 45% more satisfied with the products and a decrease in product defects up to 49% [19]. Another survey conducted in 2013 was based on basic question of addition of IoT in businesses and expectation of businesses growth with this addition. The results present that about 96% of the people believed that IoT will become an addition to a part of their businesses at some point, 38% believed that IoT will have a vital influence on their businesses, 45% responded that adopting IoT will have a positive influence on their company environment, and 63% responded that businesses which show a lack of interest in integration of IoT will be left behind [20].

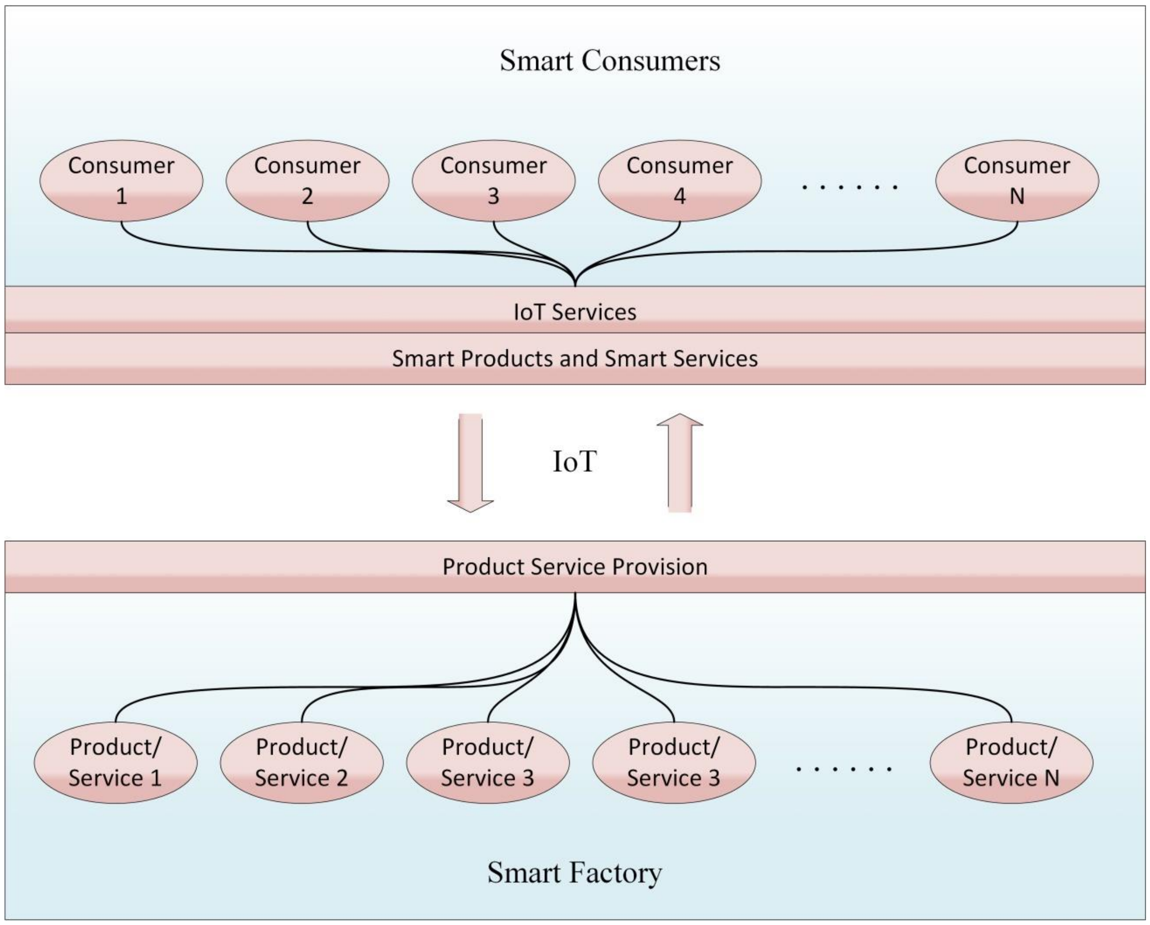


Figure 4: IoT enabled interaction between smart factory and consumers [21]

In Figure 4, an IoT technology enabled interaction among smart factory and smart factory consumers is shown. The interaction is two way, first from the smart consumers to smart factory where the users' data such as potential consumers' needs, their online behaviors and expected product behavior are passed onto the smart factory; second is from smart factory to smart factory's consumers where the smart products and services matching to the consumers' needs are delivered to them.

The Figure 5 presents an IoT based architecture for smart factory which consists of five main units as smart customer' behavior, cloud computing and big data, smart factory, smart grid and smart suppliers. The smart factory unit consists of seven main components as smart machine, smart devices, smart engineering, smart manufacturing process, data analytics, manufacturing IT and smart suppliers [16].

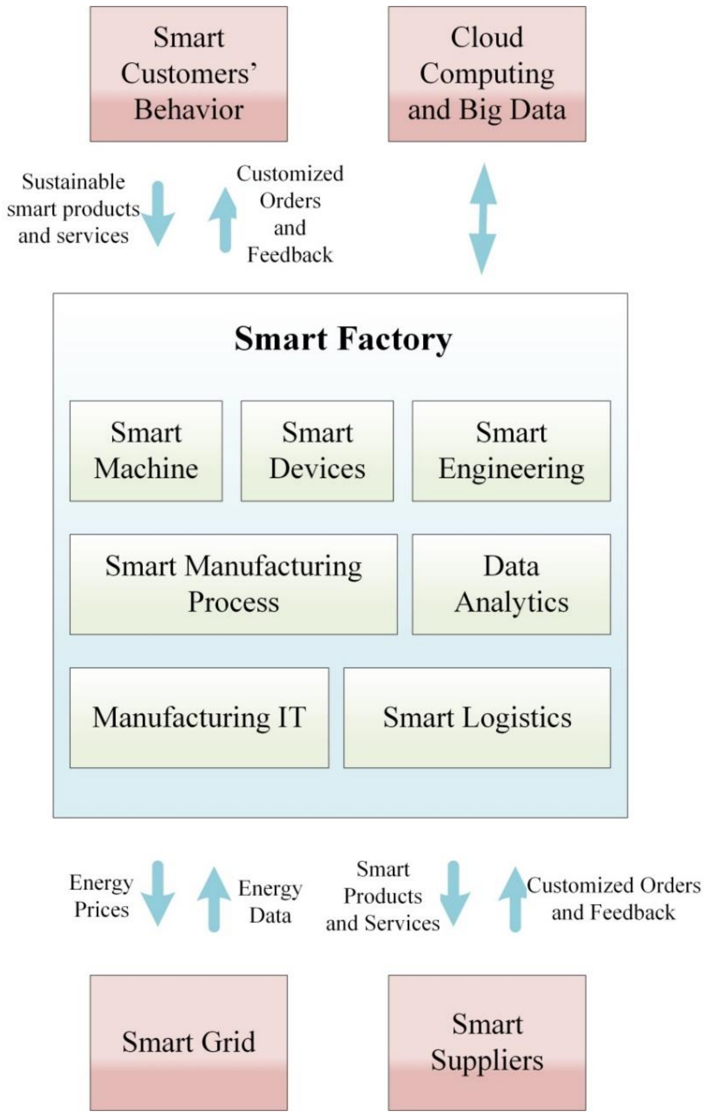

Smart Customers' Behavior
Cloud Computing and Big Data
Sustainable smart products and services
Customized Orders and Feedback
Smart Factory
Smart Machine
Smart Devices
Smart Engineering
Smart Manufacturing Process
Data Analytics
Manufacturing IT
Smart Logistics
Energy Prices
Energy Data
Smart Products and Services
Customized Orders and Feedback
Smart Grid
Smart Suppliers

Figure 5: An architecture for IoT-based smart factory [16]

The Table 1 below explains the units and components involved in the IoT based smart factory architecture.

Table 1: Units and components involved in the IoT based smart factory architecture

Unit/Component	**Description**
Smart machine	– Machine-to-Machine communication – Machine-to-Human communication
Smart device	– Field devices – Mobile devices – Operating devices – Sensing devices – Actuating Devices
Smart manufacturing processes	– Dynamic process communication – Efficient process communication – Automated process communication – Real-time process communication
Smart engineering	– Product design – Product development – Product engineering – Product production – Product's after sales service
Manufacturing IT	– Software application – Smart monitoring – Data Sensing – Automated control – Smart meters – Smart mobile devices – Intelligent production management
Smart logistics	– Logistics tools – Logistics processes
Smart suppliers	– Maximize real-time information sharing – Maximize flexibility
Smart grid	– Smart infrastructures for energy in smart factory
Big data and cloud computing	– Algorithms – Analysis of applications

Cyber-Physical Systems (CPS) can be referred as the systems with integrated computational and physical capabilities [22]. CPS plays a vital role in the development of smart factory systems, as CPS aids the managing of big data being continuously generated by the sensors, IoT networks and smart machines. Such steps will influence the machines to become intelligent and self-adaptable, hence improving the machine-to-machine and machine-to-human communication dramatically [23-24]. The integration of CPS with the factory units of production, logistics and services will transform them into Industry 4.0 factory (smart factory), with increased economic prospective [25-26].

In Figure 6, a five level CPS architecture is presented [27]. The purpose of the architecture is to list down the steps of building a CPS, in order to bring ease in CPS development, implementation and integration into smart factory. The architecture has five layers as smart connection level, data-to-information conversion level, cyber level, cognition level and configuration level. The connection level contains local data server, data-to-information conversion level contains sensing data and machine data, cyber level contains adaptive health assessments and time machine records, cognition level contains machine components along with quality check and products along with quality reasoning and configuration level contains self-optimized machine tools and self-adjustable prognostics.

The Table 2 below shows the five CPS levels along with each level’s attributes description. The main attributes of a CPS system include self-aware, self-configure, self-adjust, self-optimize, self-maintain, self-compare and self-organize.

Table 2: CPS architecture levels and attributes

CPS Level	Attributes Description
1. Smart connection level	– Self-configure – Self- adjustable – Self-optimizable
2. Data-to-information conversion level	– Integrated simulation – Remote visualization – Collaborative diagnostics – Decision making
3. Cyber level	– Twin model for smart factory components and machines – Time machine for variation identification and memory – Clustering for similarity in data mining
4. Cognition level	– Smart analytics for machine health – Smart analytics for multi-dimensional data association – Performance prediction
5. Configuration level	– Sensor network – Actuator network

Connection level establishes the connection to the local data server. Conversion level converts the sensing data and machine data to system understanding. Cyber level replicates the physical smart factories' objects, machines and involved elements and it provide the adaptive health assessments of the machine and time machine records. Cognition level performs quality check, quality reasoning for the machine and product. Configuration level performs the self-optimization to machine tools to meet the quality requirements and efficiency requirements; and it performs the self-adjustable prognostics to improve the assets life-time and to improve the product quality.

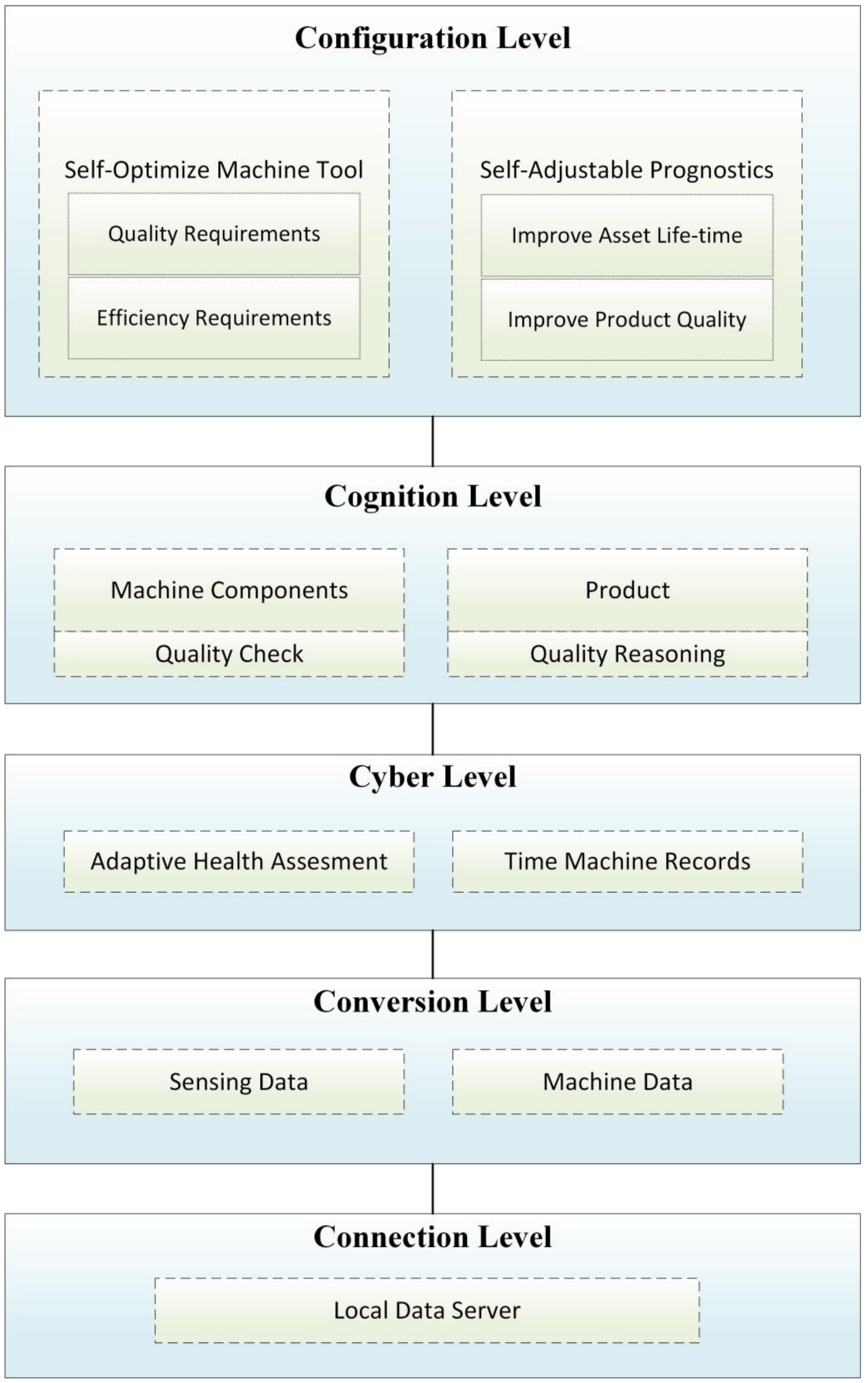


Figure 6: CPS architecture for smart factory [27]

2.2 Scheduling Mechanisms

The scheduling in smart factory involves the scheduling of tasks, processes and jobs in smart manufacturing. Also, the scheduling of smart manufacturing resources such as the sensing tasks from installed sensors, the control tasks for the smart machines and robots. The aim of scheduling in a smart factory is to optimize the production in every possible way. Scheduling mechanism is vital for the effective and optimal production of customized products in smart factory. Effective and optimal production is one which balances all involved parameters such as quality, cost, productivity, time and other system resources [28].

The smart factory solution requires a real-time scheduling approach for ordering the tasks and jobs arriving at the machines. The real-time scheduling aims to increase the factory productivity and machine utilization [29-30]. Real-time smart factory task management is achieved with the flexible event-driven reactions to the periodic/non-periodic happenings at the smart factory.

A mechanism for dynamic scheduling of services for smart factory is presented in [31]. The solution of scheduling the services of CPS is based on structure dynamics control. The adaptive scheduling and automated control are considered very crucial to the smart manufacturing based on the CPS production [32-33]. The real-time systems involve two main scheduling mechanisms as multi-pass simulation mechanism [34], and machine learning scheduling mechanism [35-36].

A multi-pass simulation is based on real-time simulation and fast mode preview of the simulation. The fast mode simulations are used to select the best scheduling policies for the shop floor control simulations on real-time [37]. Multi-pass scheduling approach can be unsuitable for real-time systems scenarios due to consuming high computational resources.

The machine learning scheduling mechanism is more suitable for real-time systems. The machine learning based mechanism requires building a knowledge based first, by running simulations based on training examples. The building phase of enough knowledge base is a time taking task. Once the knowledge base is built, it helps the simulations to acquire the results in a more robust and efficient manner. The knowledge base aids in making real-time decisions based on smart manufacturing operational constraints and parameters [38].

Main algorithms used for building knowledge base in machine learning scheduling mechanism are ANNs [39], support vector machines [40] and decision tree learning [41].

An agent cooperation mechanism for decentralized scheduling is proposed in [42]. The proposed mechanism focuses on decentralized scheduling based on agents such as order agent, product agent and resource agent. The order agent generates process execution knowledge, product agent generates production knowledge, and resource agent generates process knowledge. The agent has three main functions as time budget utility function (TBU), pair compatibility utility function (PCU) and network utility function (NCU). TBU aims to maximize the chance of executing a task before its deadline. PCU aims to enhance the utilization of a machine pair ordered together for jobs execution. NCU aims to maximize the network utilization based on pairs ordering and jobs allocations. The resource agent executes the instances using genetic algorithm for optimization process to ensure the maximization of system goals.

Some of the traditional scheduling approaches for real-time and non-real-time systems are first in first out (FIFO), shortest job first (SJF), highest priority first policy, least laxity first (LLF), modified least laxity first (MLLF), round robin (RR), earliest deadline first (EDF) and deadline monotonic (DM) [43-50]. Some of the customized scheduling approaches for real-time system scheduling are maximum urgency first (MUF), time-stepped load balancing (TLS), smoothed least laxity first (sLLF), procrastination scheduling and hybrid scheduling approaches [51-66].

2.3 Prediction Mechanisms

Prediction mechanism are very crucial in smart factory as they widely help in improving the product quality and customers experience based on learnings from past trends. The implementation of analytics tools to predict the production and consumer patterns plays a vital rule. Figure 7 shows the framework for the predictive manufacturing system [6].

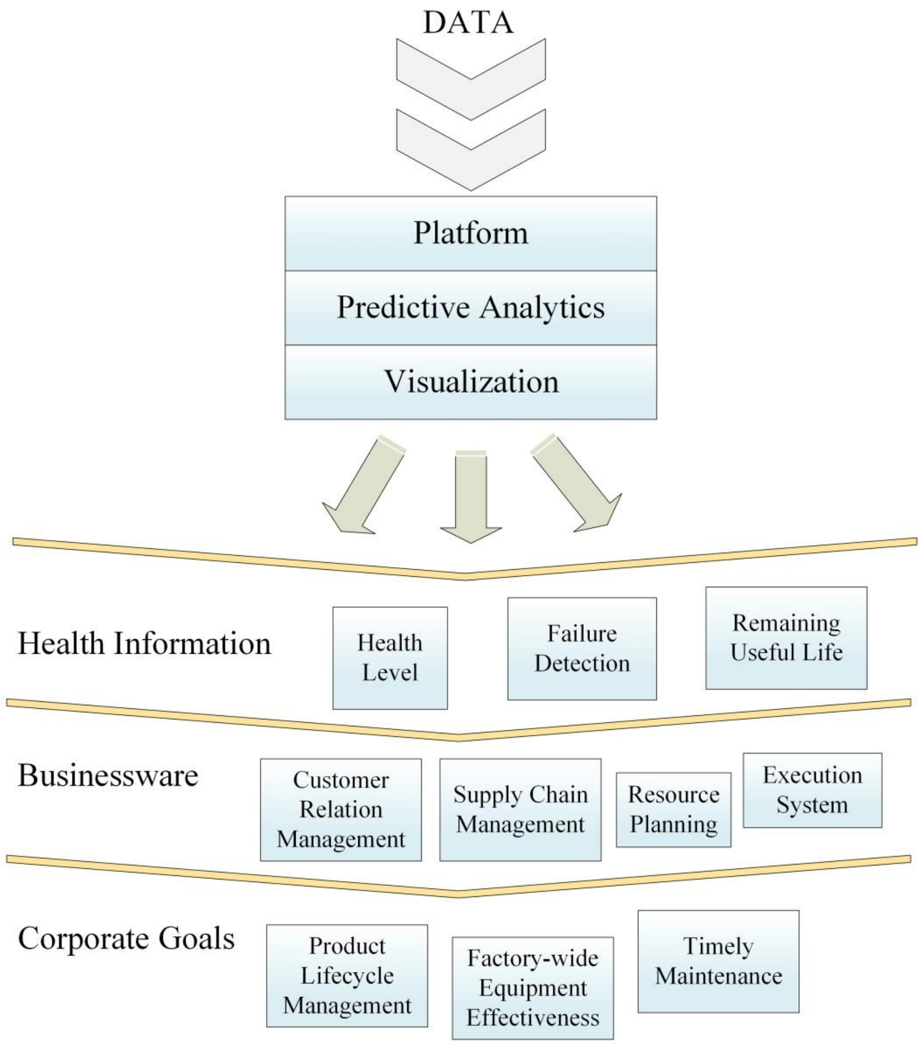


Figure 7: Predictive Analytics in Smart Factory [6]

Algorithms which focus on finding efficient and quick solutions (approximate solutions) to a problem by forfeiting the accuracy and optimality are known as Heuristic algorithms.

Algorithms which create a statistical or probability based model for the input data are known as statistical algorithms. Table 3 shows the list of heuristic and statistical algorithms.

Table 3: Heuristic and Statistical Algorithms

Heuristic Algorithms	**Statistical Algorithms**
Artificial Neural Networks	Linear/Logistic Regression
Support Vector Machines	Naïve Bayes Classifier
Genetic Algorithms	K Means Clustering
Swarm Intelligence	Support Vector Machine
Simulated Annealing	Markov chains
-	ARIMA

Prediction mechanism is used at multiple levels in the smart manufacturing. It is used for performance predictions of the system [67]. Prediction approaches are also used to predict the health conditions of smart factory tools such as a study in [68] uses ANNs based predictions, support vector machine based predictions and random forests based predictions for the tool wear predictions in the smart manufacturing. One of the major roles of prediction approaches in the smart manufacturing is of predictive maintenance [69]. Predictive maintenance refers to the timely predictions for the smart factory's equipment downtime and failure in order to improve the productivity and minimize the production cost. The study in [70], presents a baseline predictive maintenance solution which consists of components such as a target device (TD), device health index (DHI), and remaining-useful-life (RUL) predictive model. The system gets related process data and target device data as input and outputs the device health index and device's remaining useful life indicating whether device is in safe state or risk state.

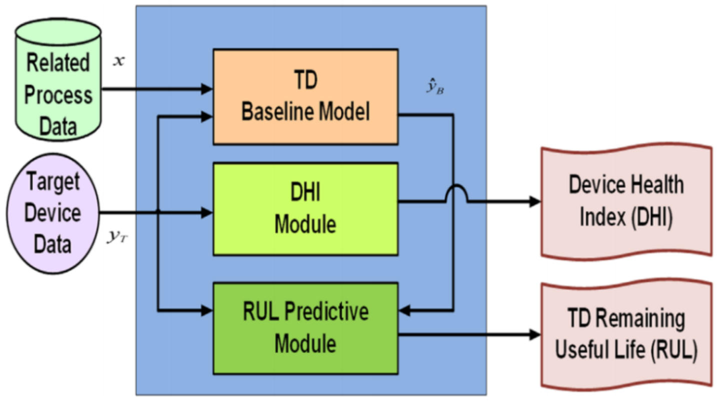


Figure 8: Baseline Predictive Maintenance [70]

The work presented in [71] combines the baseline predictive scheme with a cyber-physical agent and adds an advanced manufacturing based on cloud of things to implement a system that provides factory-wide equipment maintenance with hundreds of machines active in the smart factory. The goal is to provide a factory-wide predictive maintenance system. The study presented in [72] also proposes a cloud based predictive maintenance solution to aid the smart factory production. Many prediction mechanisms focused on task completion, time management, self-adaptive task scheduling, task replication, low-power task scheduling etc. presented in other related studies [73 - 83].

2.4 Optimization Mechanisms

In the Figure 9, the classification of optimization algorithms into two main classes as systematic optimization and heuristic algorithms is given [84]. The systematic optimization is further classified into two sub-classes as mathematical optimization and combinational optimization. The heuristic algorithms are further divided into three main sub-classes as bio-inspired optimization algorithms, hybrid optimization algorithms, and stochastic optimization. Each sub-class has a number of algorithms residing in it as given in the Figure 9.

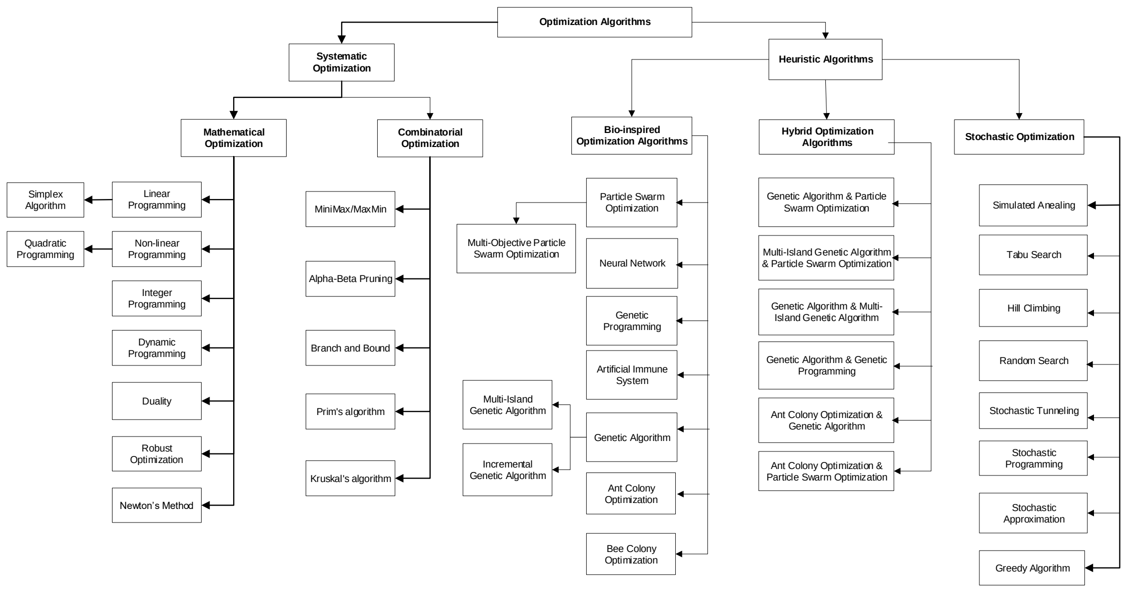


Figure 9: Typical classification of primitive optimization algorithms [84]

The task scheduling simulations when guided by simulation optimization approaches result in improved productions. The optimization model is used to generate the possible solutions which maximize the productivity and minimize the cost [85]. The work presented in [86] highlights the challenges and limitations of the process manufacturing in the current petrochemical industry. The study stresses on the adaptive smart optimization for manufacturing process and supportive control and optimization for factory-wide process production.

Industrial RFID performance evaluation is performed in the study presented in [87], for optimal design of smart factory processing. The study presented in [88], presents a review of the 3D printing by highlighting the concerns to be addressed for optimized 3D printing application. Optimization is used both at individual unit level and also with integration of all units to optimize entire systems bigger goal. The work presented in [89] suggests the integration of all existing process control and related components can result in an optimized system with minimized cost and improved energy productivity. A workshop conducted in 2014, ASCPM, highlighted the need for contrasting smart manufacturing with digital manufacturing with design modeling as its core. Design models aim to optimize the product, optimize the chain supply, and optimize the decision-making process, optimize scheduling and optimize control.

In the past many factories have been only concerned about forward logistics. The term forward logistics refers to the delivery of product from suppliers to consumers. The bulk return of products shifted the focus to reverse logistics which refers to the delivery from consumer to the factory. The work proposed by Fang et al. focuses on the reverse logistics techniques [90]. The proposed solution is an integrated IoT based three staged model. The proposed model aims to optimize the factors such as procurement, production and product recovery, pricing and strategy of return acquisition. They model considers three ways of handling products' return as refurbish the existing product, reuse the individual components of product into something or dispose the existing product. The model of optimization is built using PSO algorithm based on two heuristic methods and the proposed solution is verified and evaluated via use case scenario.

The study in [91] presents a multi-stage synchronized production system based on collaborative environment under IoT environment. With the industrial revolution, mass customization has been on rise with every passing day. The product details and specifications keep changing from one order to the next. With these changes, many other involved factors also change in the production mechanism and all these count as changing production dynamics. The proposed model aims to build a control infrastructure for the production system, which can handle the production dynamics well; keeping the components of system synchronized and optimizing the production process.

Internet of Manufacturing Things (IoMT) is the field developed by integrating the IoT and manufacturing. Another term used for IoMT is smart manufacturing. The goal of IoMT is to enhance the manufacturing experience by supporting real-time interoperability, real-time product tracking, optimized production control and execution. A detailed insight on optimization of manufacturing system is given in the book written by Zhang at al. [92].

Ghashghaee at al. conducted a study in [93] which is primarily focused role of IoT in process optimization. The study highlights the issues faced by the manufacturers and proceeds to suggest the solutions for the highlighted issues. The study focuses on IoT technologies and how the IoT technologies are used for process optimization. The challenges faced by the manufacturing systems are studied in [94]. It focuses on the manufacturing system based on IoT technologies, process management and optimization. The work presented in [95], aims to solve business optimization problems with the use of optimization algorithms such as accelerated PSO and support vector machine (SVM). The proposed mechanism APSO-SVM is used for production optimization and then also used for predicting income and project scheduling. Many other related studies have focused on optimization problems in manufacturing processes [96-107].

The study in [108] carries a survey on application of genetic algorithm broadly used for optimization. In [109], a survey for another broadly used optimization algorithm ant colony optimization (ACO) is presented. In [110], a survey for simulated annealing is presented which is also widely used for optimization. A review and analysis on the optimization algorithm particle swarm optimization (PSO) is given in [111].

2.5 Limitations of Existing Solutions

With the detailed literature review of the smart factory, now we highlight the main challenges and limitations of the proposed solutions. Table 4 below explains the existing challenges of the smart factory domain. Existing challenges of the smart factory include as environmental hazards, energy consumption, improve productivity, reliable infrastructure, and ease of integration.

Table 4: Smart factory challenges and description

Challenges	**Description**
Environmental Hazards	Environmental hazards include the states and events involved the smart factory which has the potential of adverse effects on the natural environment, people and surrounding. Solutions which neglect the green world obligations might be hazardous to environment.
Energy Consumption	Minimizing the energy consumption is very vital to smart manufacturing. – High energy consumption can lead to high manufacturing costs.
Improve Productivity	Productivity improvement t smart factory refers to – Improving production efficiency – Improving production effectiveness. – Improving product quality
Reliable Infrastructure	Reliable infrastructure is achieved through right implementation of – Predictive analysis techniques – Optimization techniques
Integration	Individual components should be independent and easy to incorporate and amalgamate.

In Table 5, we present the possible solutions to the challenges based on the literature review.

Table 5: Smart factory challenges and solutions

Challenges	Solutions
Environmental Hazards	Predictive analysis of historical data and optimize future processes to minimize environmental hazards
Energy Consumption	Predictive analysis of historical data and optimize future processes to minimize energy consumption
Improve Productivity	Efficient task scheduling mechanism
Reliable Infrastructure	Predictive analysis and optimization techniques based to minimize machines’ downtime; minimize errors; and efficient scheduling to increase response time
Integration	Layered framework solution based on multiple independent modules.

In Table 6, we present a detailed comparisons analysis of the proposed solution to some of existing related works based on sub-categories of task management and scheduling. The literature review for the task management and scheduling can be divided into five main sub-categories as

- Task/Process scheduling
- Task load allocation
- Predictive analysis
- Process optimization
- Decision support and coordinated control

Table 6: Comparisons among related works and proposed solution

Related Works	Task/Process Scheduling	Task Load Allocation	Predictive Analysis	Process Optimization	Decision Support and Coordinated Control
[112]	✓				✓
[113]	✓	✓			
[114]	✓	✓			
[115]	✓		✓		
[116]	✓		✓		
[117]	✓		✓		
[118]	✓		✓		
[119]	✓		✓		
[120]	✓			✓	
[121]	✓			✓	
[122]	✓			✓	
[123]	✓			✓	
[124]	✓			✓	
[125]	✓				✓
[126]	✓				✓
[127]	✓				✓
[128]	✓				✓
Proposed Work	✓	✓	✓	✓	✓

In the literature review, we have observed that most of the proposed solution focuses on either one or two of the above mentioned sub-categories. Whereas, in order to provide a wholesome solution for tasks management it very essential to focus on all the aspects involved.

In the table, the related works are shown with respect to focused aspects in comparison to the proposed solution. In our proposal, we integrate all the essential aspects for tasks management as task scheduling, efficient task allocation, use of predictive analysis, optimization and, decision support and coordinated control. We propose learning to scheduling mechanism for task management which provides an integrated solution resulting in efficient task management based on the concepts of task scheduling, task allocation, prediction, optimization and control.

2.6 Algorithms for Learning to Scheduling

2.6.1 Neural Networks (NNs)

The computational model (named as threshold logic) proposed in 1943 by McCulloch and Pitts led to the research of artificial intelligence-based neural networks [129]. Artificial neural networks started to flourish once the processing power of computers increased dramatically, as computation power was one of the key issues faced in the progress of ANNs at the initial stages [130].

Biologically inspired ANNs are known to produce most accurate prediction results [131]. ANN learning has two operational modes of training and testing, the system has a set of inputs, weights associated with the inputs, hidden layers and a number of outputs. In training, the neuron learns to decide whether to fire an output for a specific pattern or not, while in testing mode the accuracy of the learned model is determined.

The structure of a three-layer neural network is shown in the Figure 10, where we have five inputs, six hidden layers and three outputs. The working of a simple neuron can be explained by Equation 1 [132], whereby a typical neuron computes the output in the following manner:

$$a_k = f(\sum_{i=0}^{n} w_{k_i} x_i) \tag{1}$$

where, a_k is the output of kth neuron. x_1 , x_2, …, x_n are the inputs to the neuron. x_0 input is bias (b_k)assigning it $+1$ value, with $w_{k0} = b_k = 1$. wk1, wk2, …, wkn are the weights associated to each input. f is the activation function, which incorporates flexibility in the neural networks.

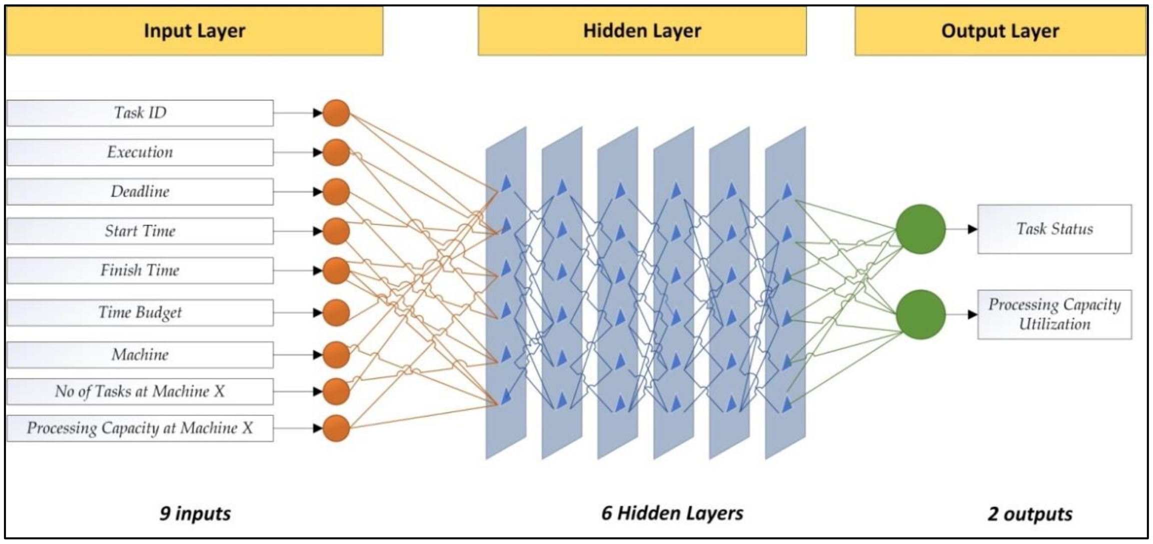


Figure 10: Neural network (NN) layers

2.6.2 Particle Swarm Optimization (PSO)

In 1995, Kennedy and Eberhart proposed PSO, which is a population-based optimization technique inspired by bird flocking and fish schooling theory and also has strong ties to genetic algorithms and artificial life [133]. In the example of the search for food by flocking birds, the bird closest to the food leads and others follow. As soon as some other bird thinks it has a food source close to it, it makes a sound and all birds start following it, changing the direction. Each particle in PSO represents a bird in the flocking example, moving at a certain velocity looking for the optimal solution in the search space.

In PSO, first a population of particles is initialized defining the number of particles that will carry the search for the optimal solution. Each particle has velocity with which it moves through the search space and fitness values. The particles move in the search space by following the particles with the best solution so far. Each particle maintains the track of two values as particle's best (pbest) and global best (gbest); pbest is the best solution achieved by the particle itself, while gbest is the best solution found by any particle in the entire population. After finding the pbest and gbest, the particle updates its velocity and position using the following equations [134].

$$v = v + c1 \times rand \times (pbest - present) + c2 \times rand \times (gbest - present) \tag{2}$$

$$present = present + v \tag{3}$$

where, v is the particle's velocity, present is the current particle position (solution), pbest is particle's personal best solution found so far in the search process, gbest is the global best solution found by any particle so far in the search, rand is a random number generated between 0 and 1, and c1, c2 are the learning factors; usually both c1 and c2 are kept 2.

Chapter 3: Proposed Learning to Scheduling in Smart Factory

In this chapter, we present our proposed task management scheme based on learning to scheduling. The proposal aims to aid the smart factory's manufacturing processes with efficient task allocation, task dispatching and task scheduling; in order to improve the overall productivity of the manufacturing process. The proposed mechanism involves the scheduling of tasks with the aid of learning modules of prediction and optimization. The learning modules provide the history based predictions and tuned parameters for improved task management.

In section 3.1 we present the conceptual design. Section 3.2 presents the hybrid ACM-FEF approach for scheduling. Section 3.3 presents the prediction mechanism, section 3.4 presents the optimization mechanism and section 3.5 presents the control mechanism.

3.1 Conceptual Learning to Scheduling Mechanism Based on Prediction and Optimization

The proposed task management mechanism is a unified solution based on machine learning and IoT technologies for smart manufacturing in smart factory. The smart factory environment is an IoT based environment with sensor and actuator networks. The sensor and actuator networks provide the contextual information of the environment to the system's machine learning modules. The machine learning based modules predict the system's states and optimize the parameters for optimal control of the smart factory's manufacturing machines and environment actuators.

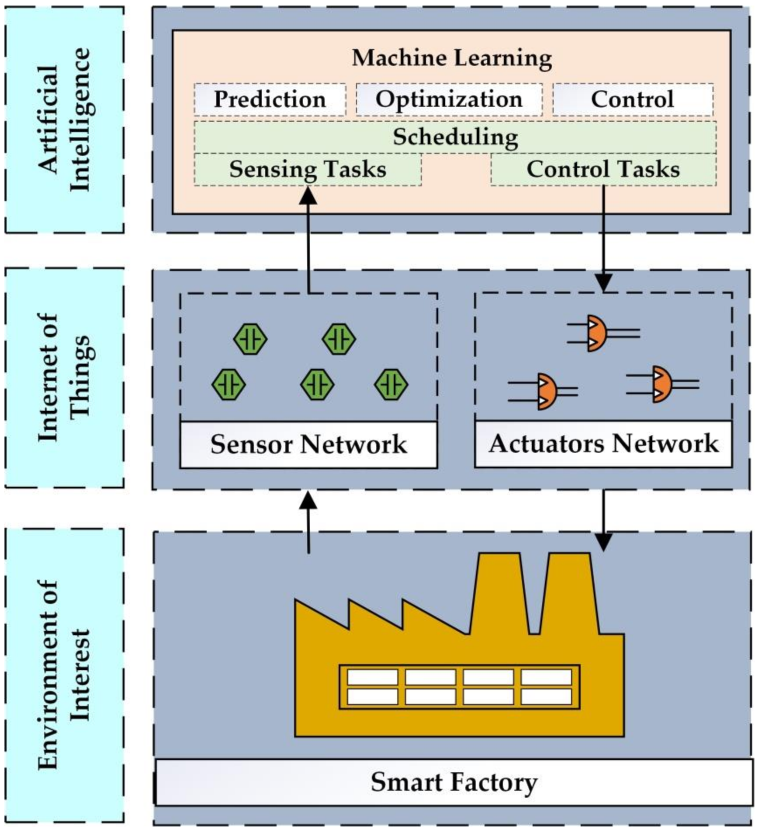


Figure 11: Layered view for proposed task management mechanism in smart factory.

We have divided our proposed task management system into four main modules i.e. (a) learning to scheduling mechanism using hybrid ACM-FEF (b) learning to prediction module (c) learning to optimization module and (d) control module. The conceptual design of the proposed system is given in the Figure 11. The smart factory has sensors and actuators installed. The sensors sense the environment and manufacturing sensing data, and pass them onto the system. The actuators are either the manufacturing machines or the environment control actuators. The actuators are controlled for executing the manufacturing processes or the environment conditions. The smart factory tasks scheduling parameters are continuously tuned using their respective learning modules of prediction and optimization.

We have developed our proposed system into five phases. In first phase, we develop our core scheduling algorithm for efficient task management, which is hybrid of agent cooperation mechanism [42] (ACM) scheduling and a fair emergency first [136] (FEF) scheduling. The hybrid scheduling scheme is referred as ACM-FEF.

In phase two, input tasks are modeled and scheduled using ACM-FEF scheduling scheme alone and tasks history data is built via multiple iterations of tasks scheduling phase. Once the history data based on ACM-FEF scheduling is collected for training purposes. Next, in the phase three, we develop our prediction module using ANN prediction algorithm. In learning to prediction, we use PSO algorithm in order to optimize the ANN weights [137]. The learned optimized prediction mechanism predicts the tasks execution status and machine utilization under the given load of the tasks based on history decisions.

In phase four, we first propose an objective function for enhancing machine utilization and then seek the optimal results based on PSO algorithm. In learning to optimization, ANN algorithm is used to tune the PSO particles positions and velocity. The optimization module aims for the maximum machine utilization in the smart manufacturing process.

In phase five, we develop the control module based on inference rule engine [138]. The control module gets optimal parameters input from optimization module and generates the actuators; control commands based on the optimal parameter settings.

At last, the scheduling modules fully implements the learning to scheduling procedure by integrating the overall modules and using the outputs of prediction module, optimization module and control module altogether. The learning to scheduling module aims for optimal machine pairing based tasks allocation; it also uses predictions based on history decisions consequences to make informed scheduling decisions.

We have proposed two improved variations of PSO algorithm named as VB-PSO and R-PSO [136]. The improved variations are used in both prediction and optimization modules to enhance the modules' performance. In prediction phase, we propose PSO algorithm based ANN implementation referred as PSO-NN, in optimization phase, we propose ANN algorithm based PSO implementation referred as NN-PSO.

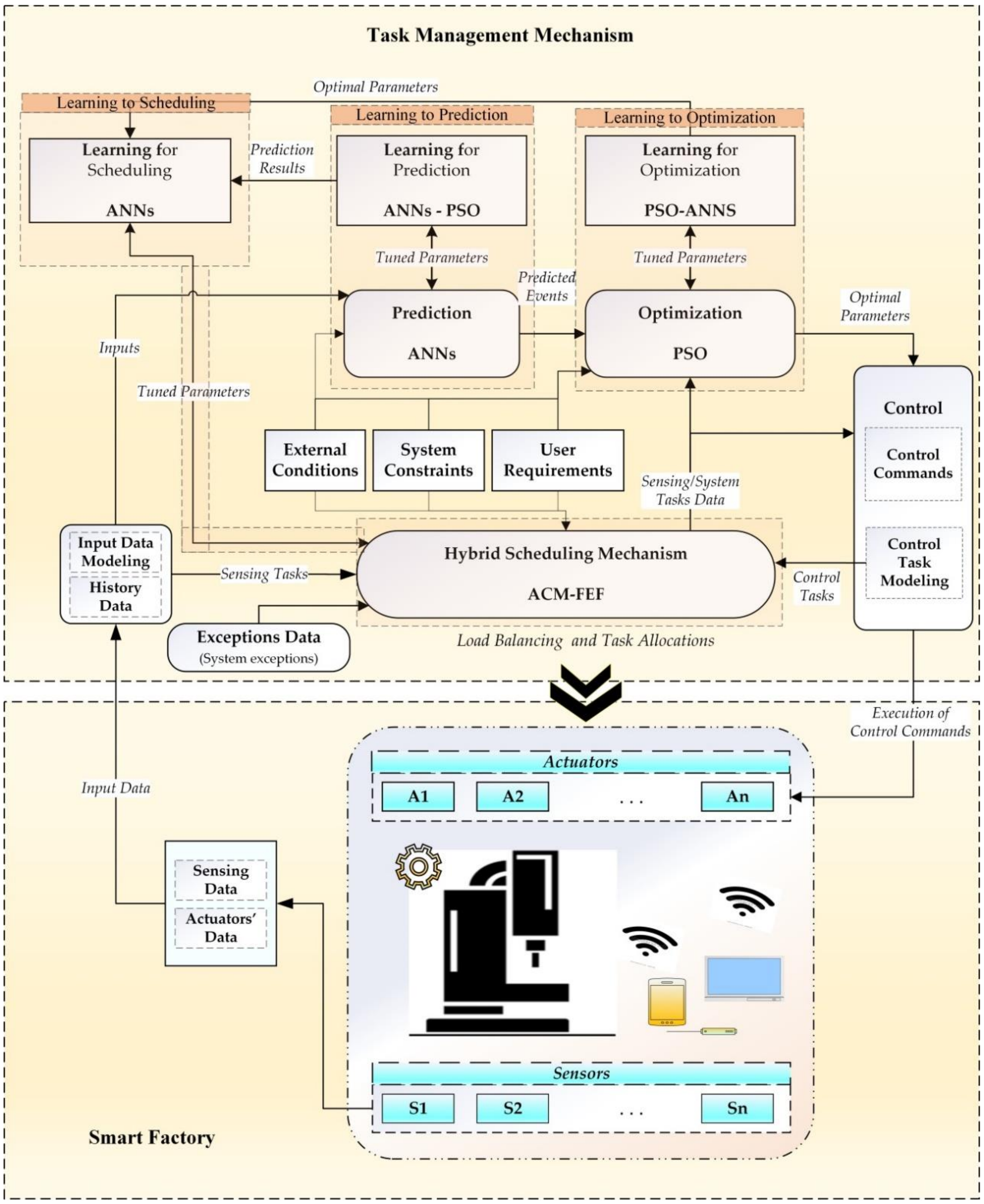


Figure 12: Conceptual design of the proposed task management mechanism based on learning to scheduling in smart factory

In Figure 12, we present the conceptual design of or proposed task management mechanism based on learning to scheduling in smart factory. The smart factory will have two types of data collected as sensing data collected from installed sensors in the environment and the actuators' data collected from installed actuators. System constraints, user requirements and external conditions can be added inputs to the system based on the scenarios. In input data modeling, the sensing tasks are modeled to be sent to the scheduler. The scheduling outputs are maintained for building the scheduling history data. The scheduler sends the sensing tasks' data to the optimization module for optimal parameter tuning. Sensing tasks' data is then sent to the control module, the inference engine in the control module generates the optimized control commands based on the inputs from optimization module. Next, control module models the control tasks to execute the control commands which are sent back to the scheduler. At scheduler, when a control task is executed, it triggers the control module to perform the control command.

3.2 Proposed Learning to Scheduling Mechanism using Hybrid ACM-FEF

In this section, we present the hybrid scheduling mechanism of ACM-FEF. The scheduling mechanism is a hybrid of two scheduling schemes as agent cooperation mechanism (ACM) and fair emergency first (FEF).

The smart factory is considered to have N machine networks, where machine networks have local and global agents following ACM mechanism. The purpose of ACM mechanism is to allocate the tasks to machine networks with an aim to maintain the maximum machine utilization among the machine networks. Next, the tasks allocated to machine networks are further allocated to the existing machines, where each machine has varying task load with varying priorities. The task load scheduling at each machine level is done following the FEF scheduling.

3.2.1 Agent Cooperation Mechanism for Scheduling

The agent cooperation mechanism (ACM) sub-module is inspired by the scheduling mechanism presented in a related study [42]. In smart factory environment, we consider multiple manufacturing machine networks. Each machine network is assigned an agent to manage the task allocation process. The goal of the agent is to allocate tasks to machines in such a way that overall productivity of the manufacturing process is increased. The system has one global agent; the task of the global agent is to maximize the machine utility among all machine networks.

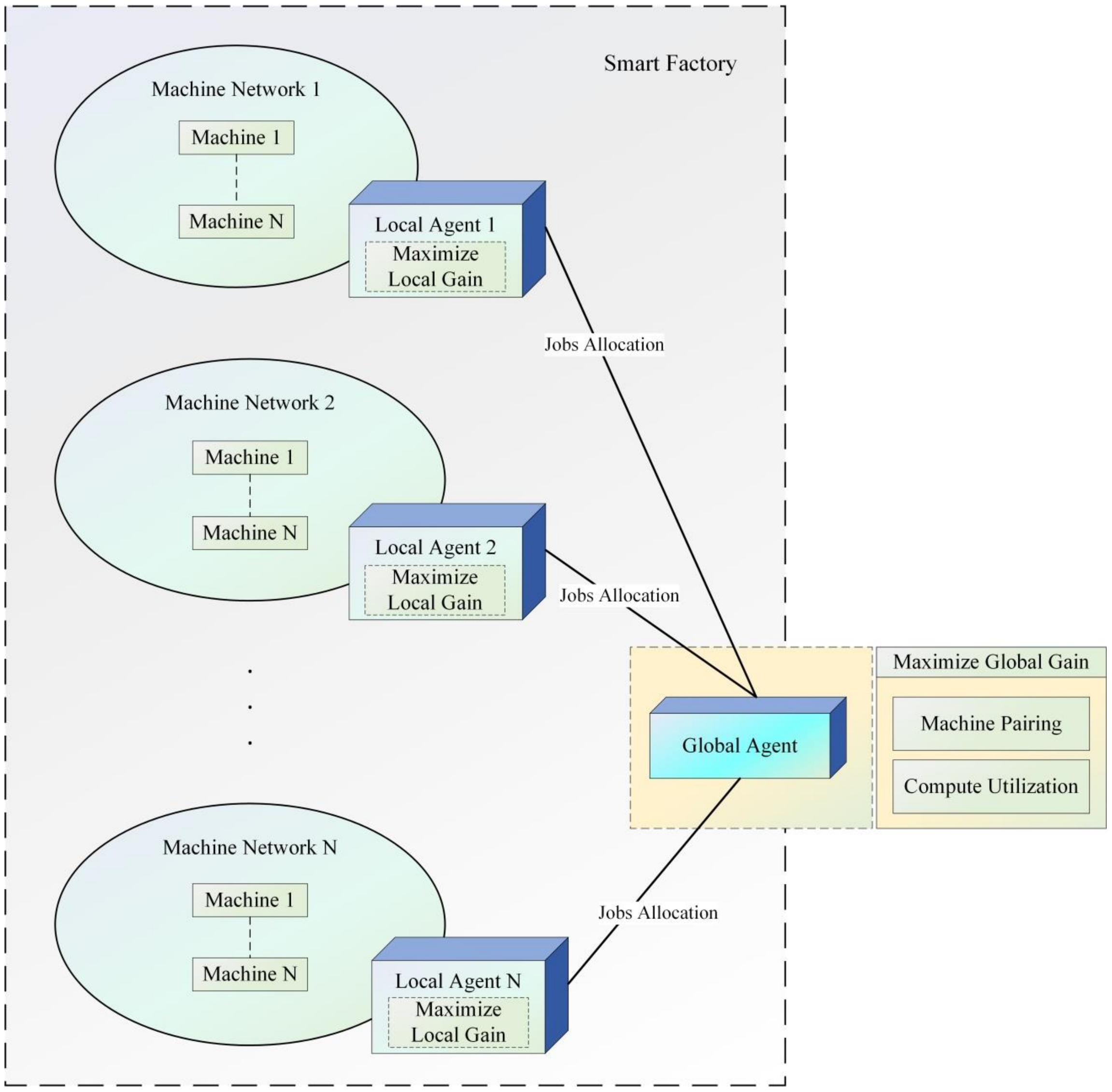


Figure 13: Conceptual diagram of agent cooperation mechanism

The Figure 13 presents the main concept of agent cooperation mechanism with N machine networks; each network having a local agent to maximize the machine utilization at its own network and a global agent to maximize the overall system's productivity. The global agent has two main functions as machine pairing and computing utilization. The global agent has complete knowledge of the system such as number machine networks, number of machines per network, and each machine's current load and capacity.

3.2.2 Fair Emergency First Task Scheduler

In this sub-section, we present the basics of FEF scheduling algorithm [136]. The FEF scheduling algorithm is designed to maximize the machine resources and minimize the tasks starvation rate.

The FEF scheduling algorithm considers the input tasks first divided into two main types as event driven tasks and periodic tasks. The event driven tasks are further divided into two subtypes as urgent event driven tasks (UET), normal event driven tasks (NET). The periodic tasks are further divided into two main types as priority periodic tasks (PPT) and normal periodic tasks (NPT). The event driven tasks are given high priority, as they might be emergency triggered and one-time tasks. The UET are considered to be of high priority, followed by NET, next priority is given to PPT followed by NPT. These priorities are not static, and the proposed algorithm allows flexible options to define and alter task priorities based on the scheduling scenarios.

The primary focus of FEF algorithm is to meet the tasks deadlines based on their priorities; in parallel, saving the starving tasks by rightly utilizing any free resources. The starving tasks can be defined as any tasks which are in waiting state for a long period, due to system priorities, load or unexpected events. The flowchart for FEF algorithm is shown in the Figure 14. First the scheduler extracts the tasks arriving at the system based on arrival times. If the current task is urgent event driven task, it is executed right away. If the task is normal event driven task, then urgency measure (UM) is checked to see whether the machine slot can be used for any low priority starving tasks or not. If not, then current task is executed else starving task is given the slot. Next, priority periodic task is checked, where failure measure (FM) is checked to see if the priority periodic tasks can wait and slot can be allocated to the starving low priority task or not.

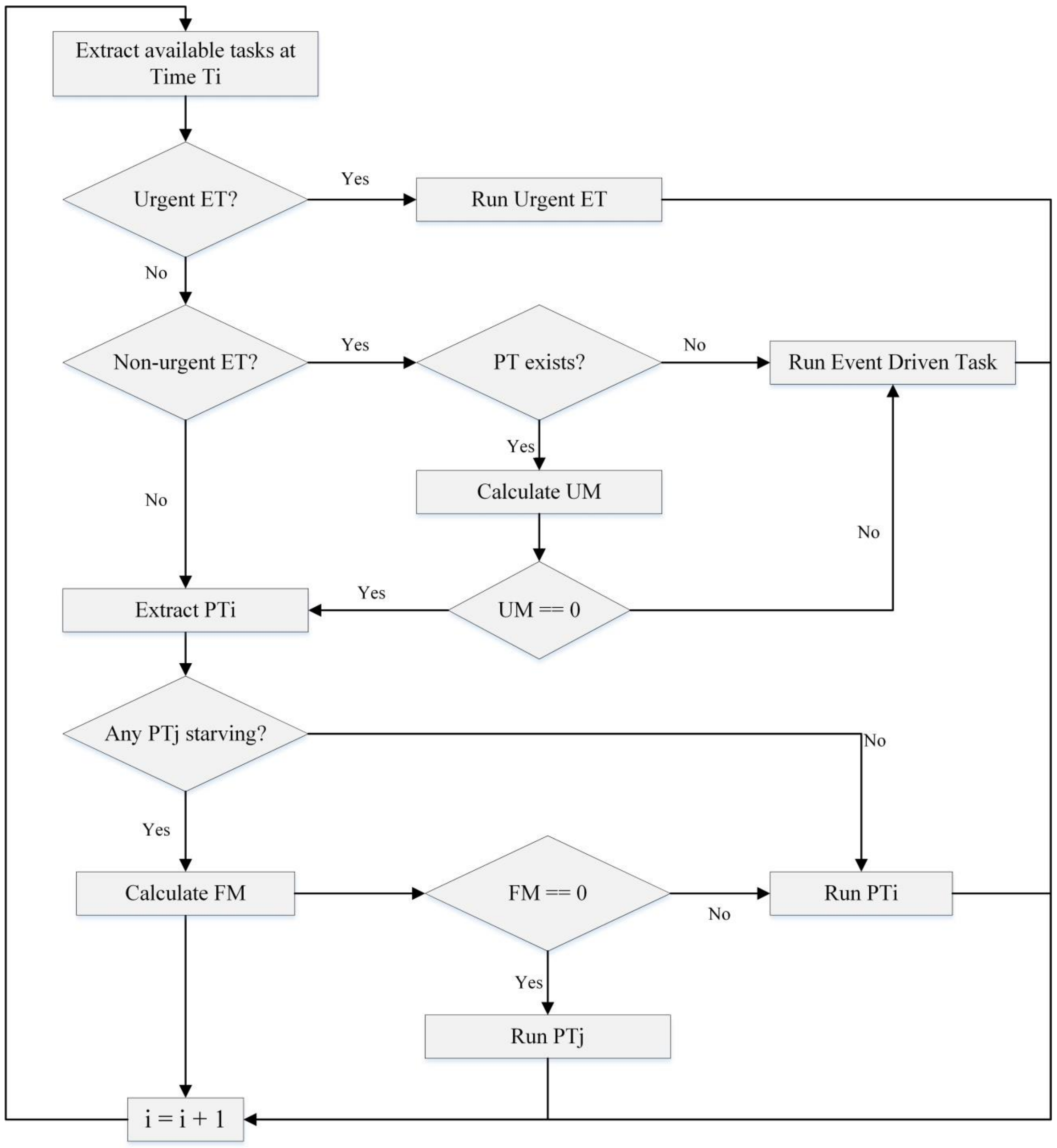


Figure 14: Flow chart for FEF Scheduling Algorithm

Urgency measure and failure measure are two learning modules in the intelligent FEF scheduler. Urgency measure is computed to decide whether at the current state the system can allocate the machine resources to a starving task, given that the current event driven task can wait to execute without damage. Failure measure is computed to decide whether at the current state the system can allocate the machine resources to the starving task, if current periodic task can wait to execute without damage. The FM and UM are the two learning factors to make respected decision using the ANN learning algorithm based on history decisions.

In order to calculate the UM, the system computes the slack between the event driven tasks and periodic tasks. Slack can be defined as the difference between the deadline of the tasks and the execution time left to finish the task. The decision of whether starving task should be given machine resources or not is made based on Equation 4.

$$Slack\ (ET) >= x \times (Slack(PT)) \tag{4}$$

Where, Slack (ET) is the slack or event driven tasks, Slack (PT) is the slack computed of periodic task and the distance between both slacks should x times; x is initially set as 2. Gradually the value of x is learned using ANN as the system runs the tasks and history data is built (Figure 15).

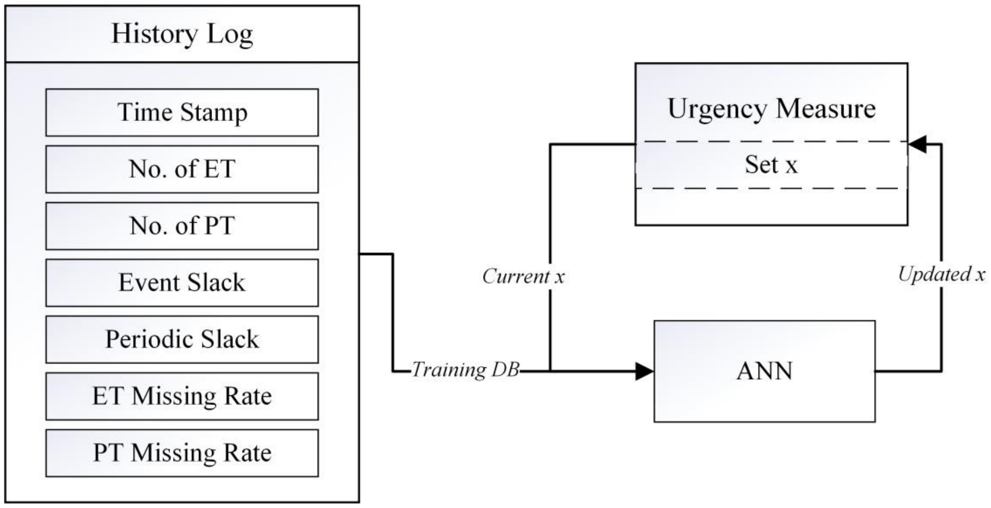


Figure 15: Learning of X in Urgency Measure (UM). ET: event-driven task; PT: Preemption Threshold

The learning factor FM is calculated among the periodic tasks of different priorities. It is calculated to make decision whether current periodic task can wait without any damage, and the machine resources can be allocated to the starving tasks or not. The history data is first built and used to make the informed decisions. The module predicts the possibility of safe execution of starving tasks and without delaying any other high priority periodic task based on learning using ANNs (Figure 16).

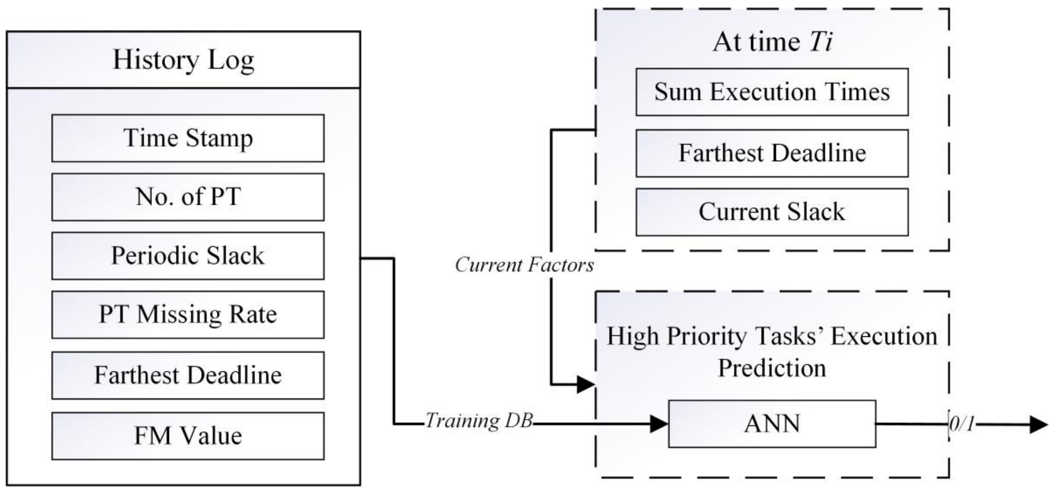


Figure 16: Prediction for high priority tasks' safe execution

Additionally, two bits as reserved bit and priority bit are added to ensure the flexibility of scheduler setting in multiple scenarios. The reserved bit indicates if any numbers of machine chunks are to be reserved during online scheduling for handling the possibility of any urgent unexpected events. The preemption bit when added to any tasks gives it the authority to halt any high priority running task and to be executed first. These two bits are to make alterations based on the scenarios.

3.3 Learning to Prediction for Scheduling in Smart Factory

In this section, we present the learning to prediction mechanism using PSO based ANN prediction algorithm. The used mechanism is published in [137], where it is used for energy predictions in smart building. In this work, we use our proposed prediction-learning mechanism in order to aid the smart factory scheduling process by predicting the tasks execution status and machine utilization.

3.3.1 Prediction using ANNs

In this sub-section, we present the ANN based prediction model. The model has nine inputs, six hidden layers and two output layers.

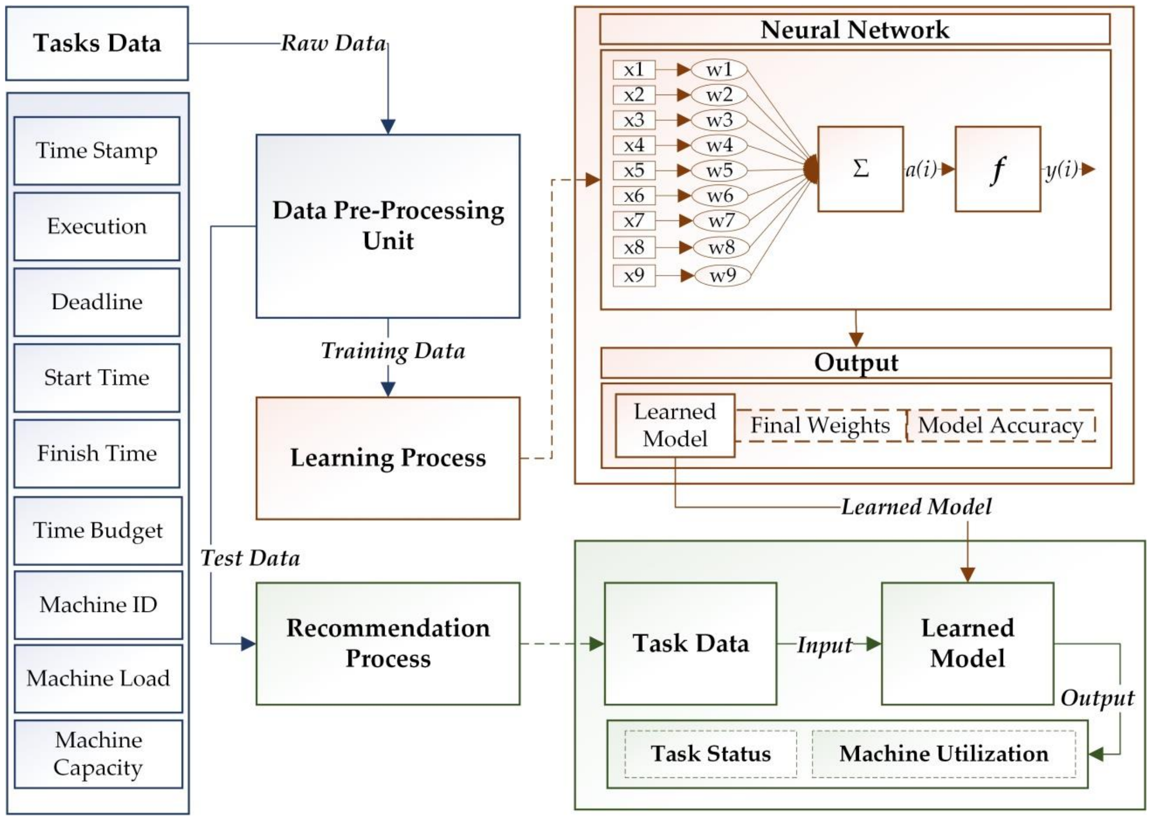


Figure 17: Prediction Model using ANNs

The system takes tasks data of time stamp, execution time, deadline time, start time, finish time, time budget, machine ID, machine load and machine capacity as input. The data is first pre-processed and then passed onto the training module where training is done based on ANN with six hidden layers. The output is prediction accuracy computed for the task status prediction and machine utilization prediction.

3.3.2 Learning to Prediction using PSO and ANNs

In this section, we describe the learning to prediction mechanism for smart factory tasks scheduling and management.

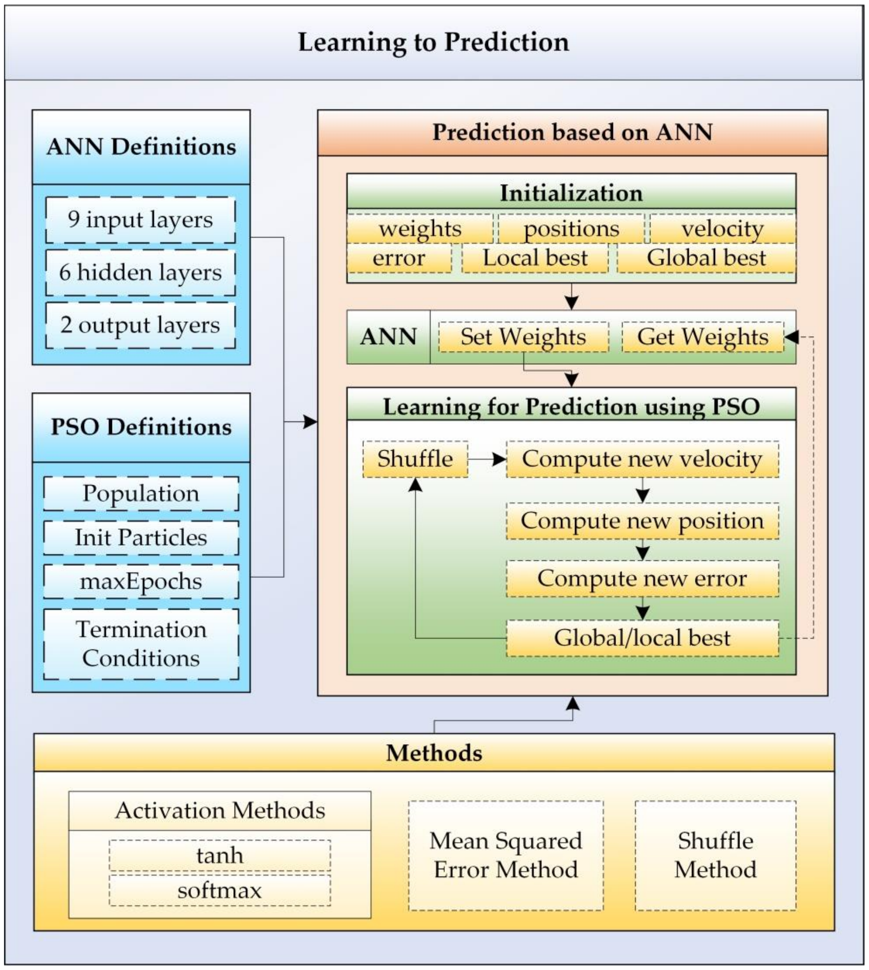


Figure 18: Learning to prediction model based on ANN

In learning to prediction, the ANN prediction algorithm's weights are learned using PSO algorithm which is an optimization algorithm. Figure 18 shows the learning to prediction configurations. Initially the input is given to the ANN learning module based on six hidden layers. The PSO algorithm is applied at ANN learning iterations for learning ANN weights. PSO algorithm takes the neural networks in ANN iterations and struggles to optimize the neural weights to achieve the high accuracy.

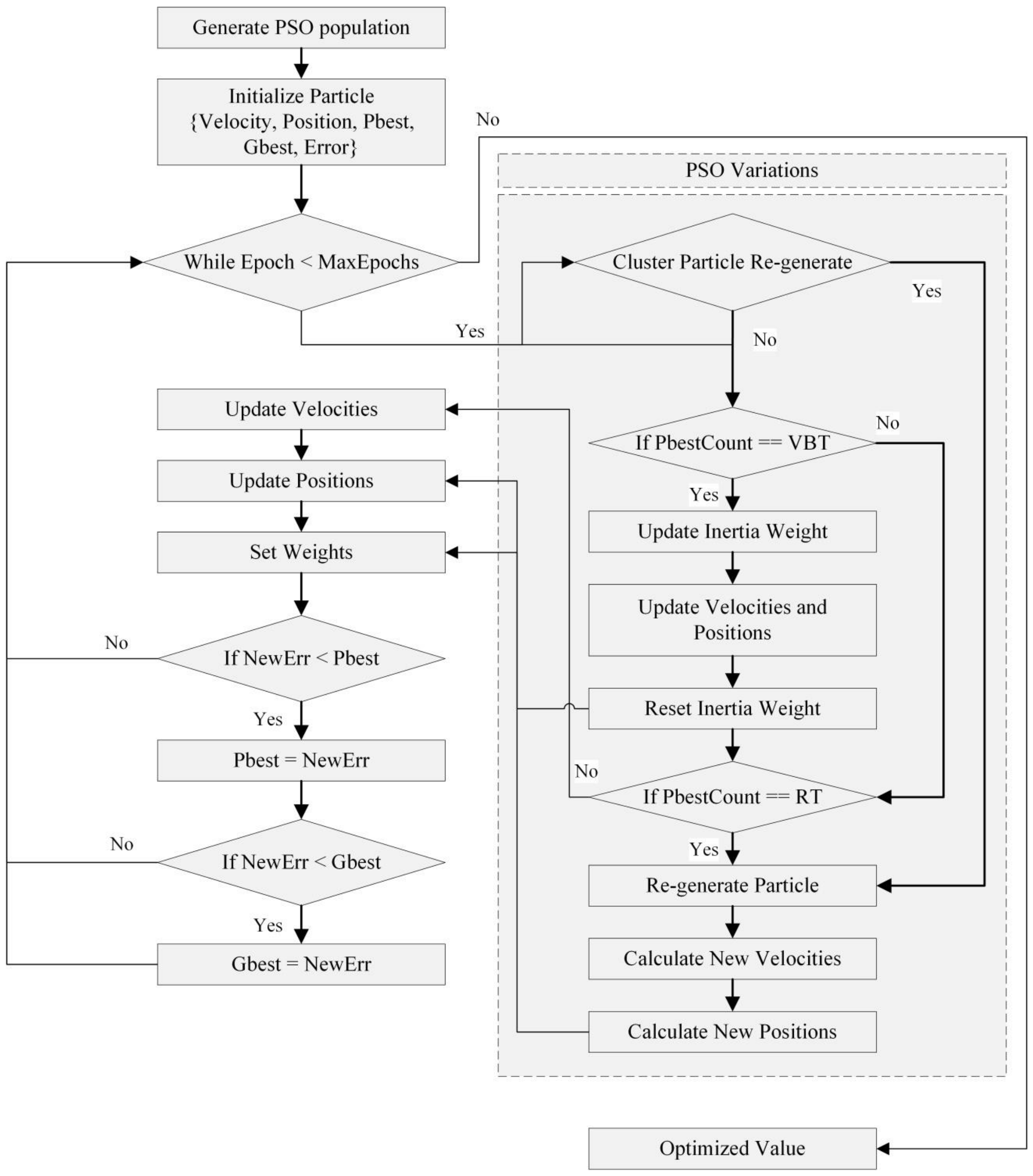


Figure 19: Flow chart for PSO variations of R-PSO and VB-PSO

The learning algorithm used to learn ANN weights is an optimization algorithm named as particle swarm optimization (PSO). In PSO, a number of particles populations (typically between 12-20 numbers of particles) are generated. Each particle in the PSO population contains two parameters as particle position and particle velocity. Initially the particle velocity and positions are initialized. In PSO iteration, the particle velocity and position are updated. Each particle maintains two values local best as Pbest, and global best as Gbest. The Pbest is the particles own best position achieved, and the Gbest is the global best values achieved by any particle in the population. Each particle position represents the ANN weights and the Gbest is the best weights found by PSO.

We use the two variations of PSO named as re-generation based PSO (R-PSO) and velocity boost PSO (VB-PSO) [137]. The R-PSO involves a regeneration threshold (RT). In R-PSO if no improvement in the Gbest is seen after a number of iterations, define by RT, then particles found in close clusters are regenerated to new random locations; in order to fasten the solution search process. The particles' distance to locate close clusters is calculated using Equation 5.

$$Inter\ Particle\ Distance\ (IPD) = c \times Number\ of\ Particles \tag{5}$$

Where, IPD is the minimum distance threshold between two particles, c is a constant value for limiting IPD set as 0.15.

In VB-PSO, a velocity boost threshold is maintained as VBT, and if no progress in the value of particle's Pbest is observed till reaching VBT then particle's velocity is boosted using the velocity change equation with new inertia weight proposed for VB-PSO. The new inertia weight is derived from the combination of constant inertia weight shown in Equation 6 and random inertia weight shown in Equation 7. The new inertia weight is shown in Equation 8.

$$Contant\ Inertia\ Weight = c1 = 0.7 \tag{6}$$

$$Random\ Inertia\ Weight = 0.5 + \frac{Rand()}{2} \tag{7}$$

$$New\ Inertia\ Weight\ (W_i) = c1 + \frac{Rand()}{3} \tag{8}$$

Where; c1 = 0.801. The detailed mechanism of proposed R-PSO and VB-PSO algorithms can be referred in our published work in [137].

3.4 Learning to Optimization for Scheduling in Smart Factory

In this section, we present the learning to optimization mechanism for scheduling in smart factory. First, we make the smart factory environment assumptions. Table 7 provides the list of parameter assumptions for the smart factory environment. There are N numbers of machines, X number of machine pairs and K number of tasks coming at each machine.

Table 7: Sensing tasks parameters for candy box factory

Task Parameters	**Parameter Description**
$M_1, M_2\ldots, M_N$	N number of manufacturing machines
$P_1, P_2\ldots, P_X$	X number of machine Pairs
$T_1, T_2\ldots, T_K$	K number of tasks at each machine
TID	Task Instance ID
TJ	No of Jobs required by a Task
$TID(T_{J,b})$	Operation of task TID at position b
TB_J	Time Budget: deadline of an operation in a task
FT_J	Proposed finish time for operation
LFT_J	Latest possible finish time
ST_J	Proposed Start time
E_J	Execution time of operation
IT_P	Idle time for a pair

The finish time is computed by adding the execution time to the start time. Time budget is the difference between least finish time and finish time.

$$FT = \text{ST} + \text{E} \tag{9}$$

$$TB = \text{LFT} - \text{FT} \tag{10}$$

3.4.1 Optimization Objective Function for Machine Utilization

In this sub-section, we present the objective function to be optimized for scheduling. The optimization module aims to increase the machines utilization with the help of three functions as time budget utility (TBU), pair compatibility utility (PCU), and network compatibility utility (NCU) [42]. The TBU functions aims to maximize the tasks possibility of execution before its deadline. The PCU function makes sure that jobs involved in a task are assigned to machine pairs with maximum utilization. The NCU function looks for the overall networks utilization, making sure that tasks are being assigned to all the machines in a network, in such an order to increase network utilization.

These three functions of TBU, PCU and NCU are used to make an objective function (OF) for the scheduling in smart factory as

$$Utilization\ OF = (\alpha)(TBU) + (\beta)(PCU) + (\gamma)(NCU) \tag{11}$$

Where, α β, and γ are the weights of the functions TBU, PCU, and NCU. The objective function aims to maximize the α, β, and γ values to maximize the machine utilization.

The optimization module aims to pair machine combinations for task execution with maximum machine utilization. Extract all possible pairs. Calculate Utilization factor for each extracted pair. Allocate tasks to pair with maximum utilization. The optimization algorithm used is PSO. The Figure 20 shows the optimized scheduling mechanism based on PSO algorithm.

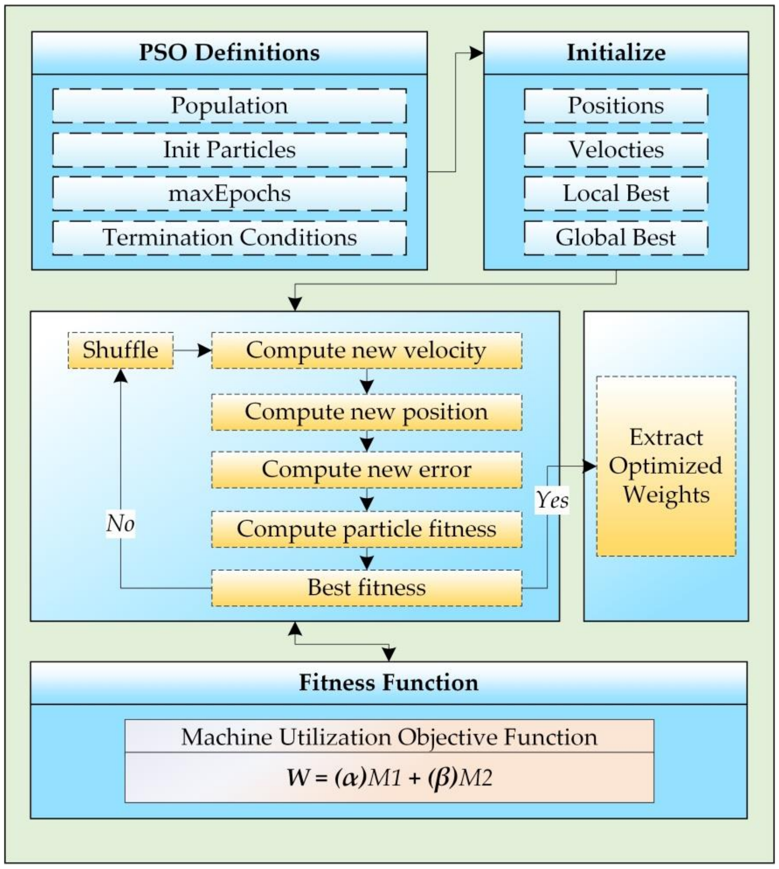


Figure 20: Optimized scheduling mechanism for maximizing machine utilization

At first the PSO population is defined and initialized. The aim of PSO particles is to find positions where machine utilization weights are at maximum for each machine and overall network. The fitness function is defined as the machine utilization function based on number of machines in a smart factory scenario at a given time. Once the global best values are found, the module returns the maximum machine utilization weights to tune the machine utilizations settings accordingly.

3.4.2 Learning to Optimization using ANNs

In this scenario we present the learning to optimization mechanism. We add ANN based learning to optimization module. It takes history PSO data as input for preparing the learning model. The history PSO data includes the particles values of positions and velocities along with errors, and Pbest and Gbest. Initially the tasks' scheduling is done based on optimization module without learning to build the history data logs.

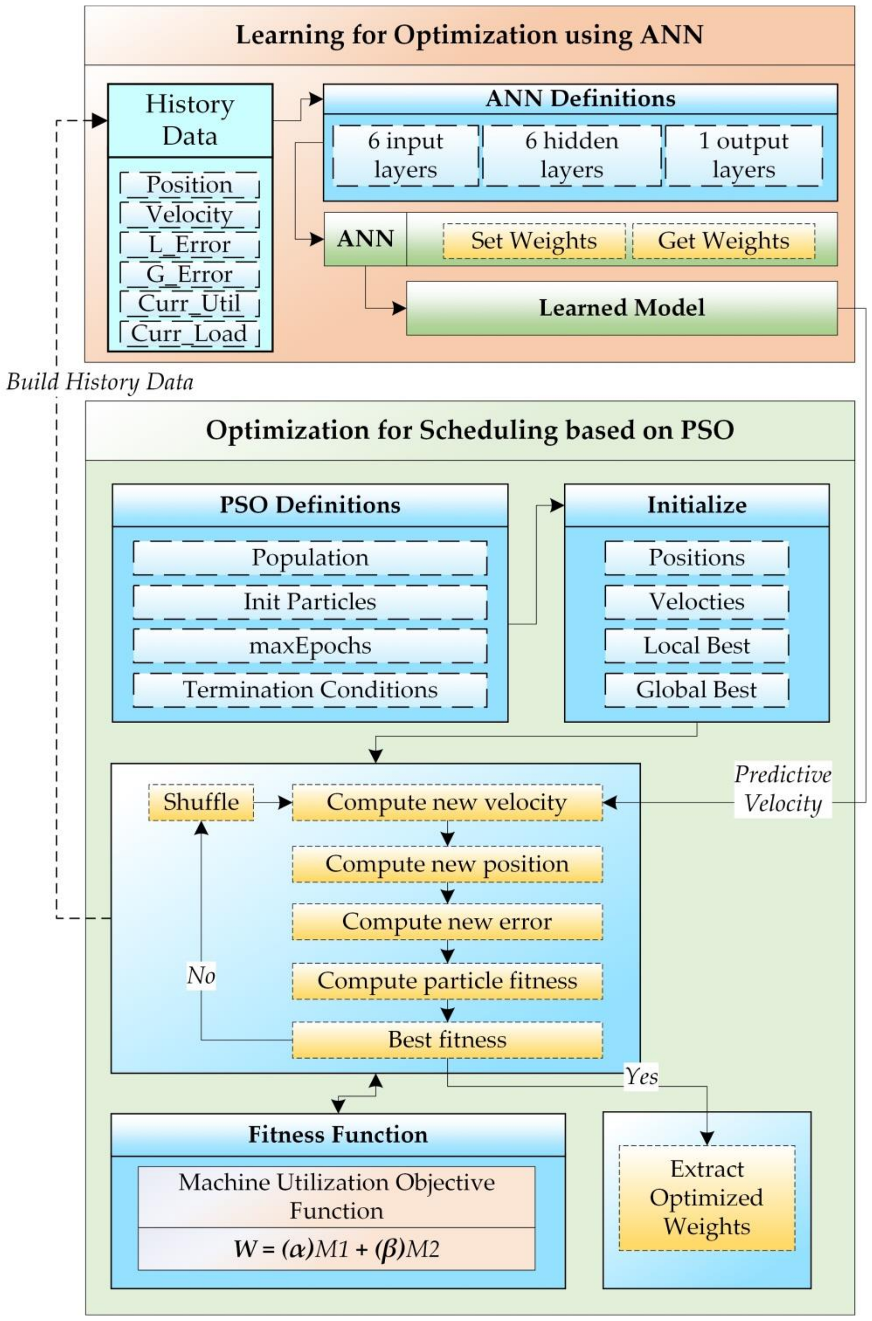


Figure 21: Learning to optimization based on PSO for scheduling

Once the history data is built then it is preprocessed and passed onto the learning module of ANN to prepare the learned model which is further used to make the predictions for PSO velocities and positions.

While the PSO algorithm runs and particles strive to find the optimal solution; in PSO iterations the next velocity of the particle is tuned based on the learning from the history logs. The ANN model has four input parameters such as particle, velocity, error rate global, and error rate local. It has six hidden layers and two output layers (Figure 21).

3.5 Control Mechanism for Scheduling in Smart Factory

In this sub-section, we present the control mechanism for the control tasks generation and autonomous machine control in smart factory. The control tasks are generated based on scenario thresholds and conditions. In this section, we use one of our published works' inference engine modules [138] and alter it to fit the smart factory scenario.

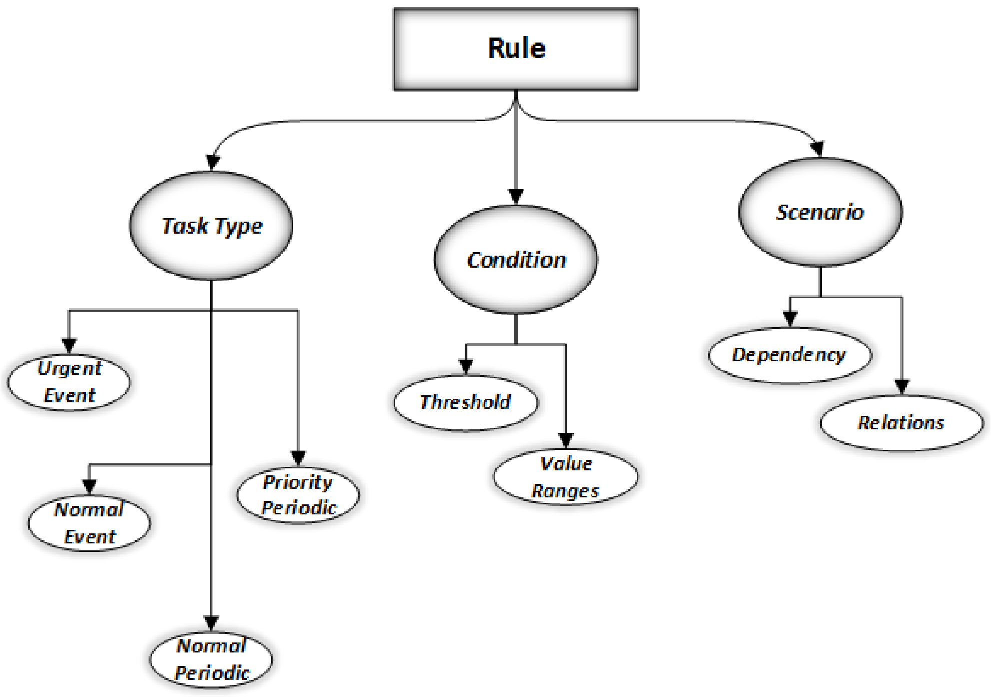


Figure 22: Defining of a rule in inference engine for control task generation

The inference engine is where all the smart factory rules are listed along with the thresholds and if-else conditionings. The inference engine rules comprise of arriving task type, condition associated with the task and the contextual scenario of the task (Figure 22). The conditions in rules contain the threshold values, which if met then the rules are fired.

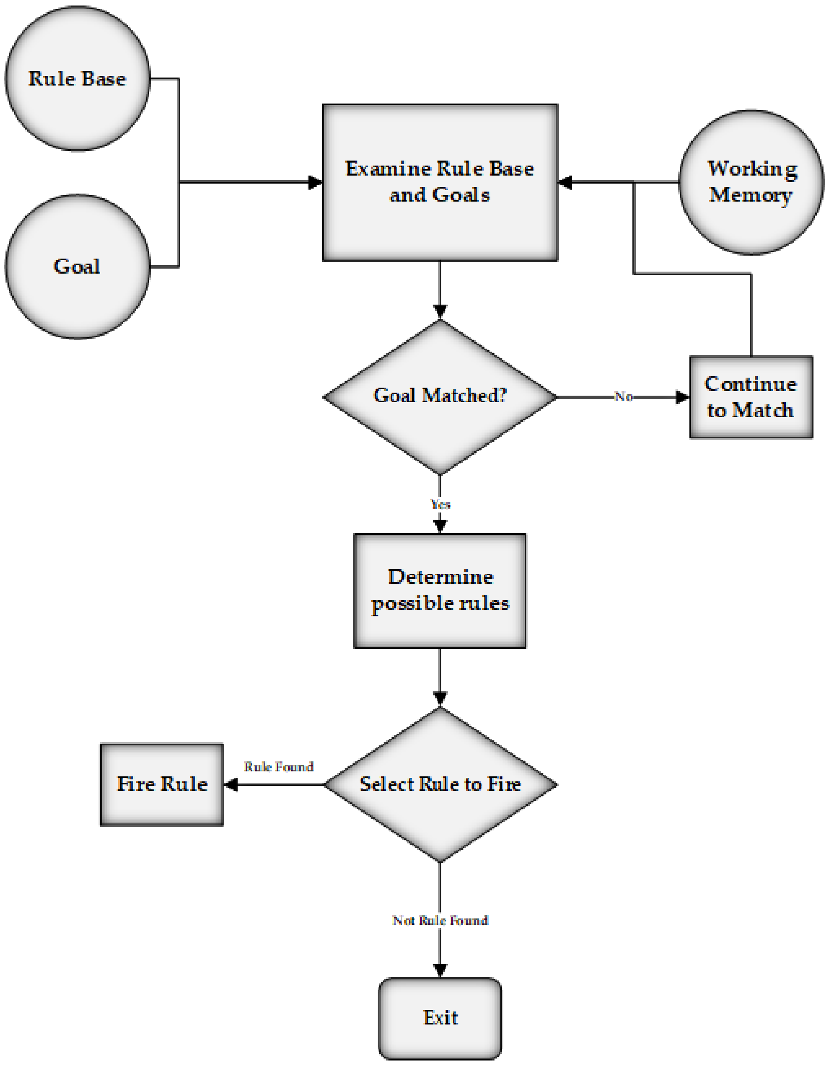


Figure 23: Workflow of firing a rule in inference engine for control task execution

Figure 23 shows the detailed flow chart for firing a rule in an inference engine. First all the rules are extracted and when a new sensing or system task value is arrived at the inference engine, the value is checked for all existing rules. The value can meet one or multiple rule conditionings; each rule conditioning when met, the rule is fired. By firing rule, it means to generate its response.

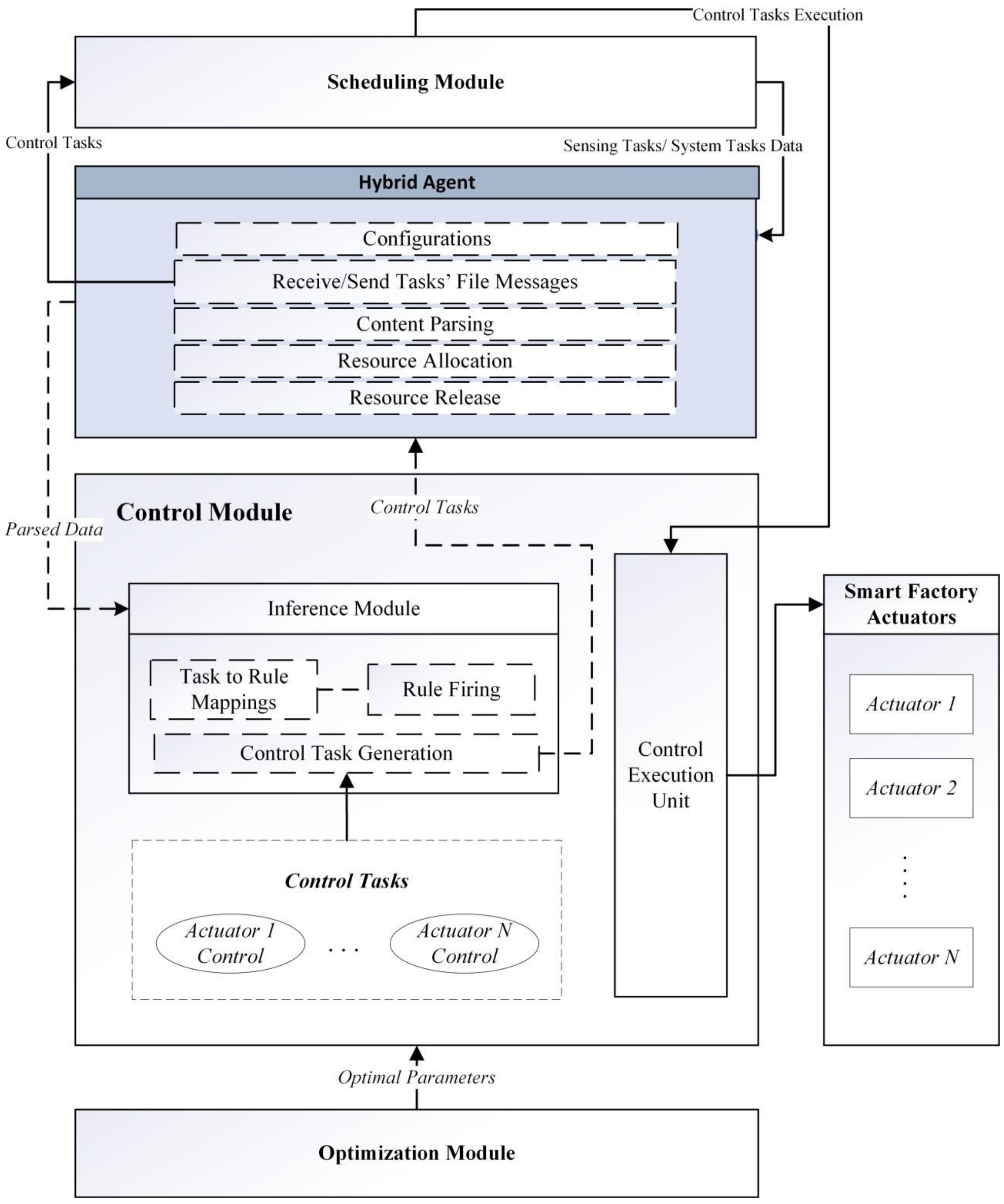


Figure 24: Control module interactions and working

The Figure 24 shows in detail the interaction between scheduler, control modules and system actuators. The scheduler sends the sensing tasks and system tasks data to the control module via an agent referred as hybrid agent in [138]. The agent parses the received data and sends the meaningful content to the inference module. At inference module, the received tasks data is mapped onto the rules and once the conditioning is met the rules are fired. Once the rule is fired the control tasks are modeled and sent to the scheduler to be executed at scheduler. When the control tasks are executed at the scheduler, the control command is sent to the control unit which executes the control commands at the actuators.

Chapter 4: Simulation Developments for Learning to Scheduling Experiments

4.1 Environment Modeling

In this sub-section, we present our smart factory environment modeling for the simulations and experimentation. The smart factory has sensors and actuators installed onto the site to execute the production tasks and services.

The sensors installed are of two types as ambient sensors and on-machine sensors. The ambient sensors are installed into the surrounding of the smart machine to get the surrounding environment sensing values. The on-machine sensors are installed onto the machine to get the machine related sensing values.

The sensors are installed with a motive of sensing the environmental conditions, sensing the manufacturing process conditions, sensing the factory emission conditions, monitoring the machine health, and sensing safety conditions. The roles of the sensors vary based on their installment scenarios and locations. The types of ambient sensors and on-machine sensors installed can be as shown in Table 8 depending on the selected smart factory scenarios. The main sensor categories are environment sensors as temperature sensor, light sensor, humidity sensor, sound sensor and vibration sensor. Monitoring sensor to read scenario based values as flow sensor, gas sensor and acceleration sensor. Detection sensors include motion sensor and occupancy sensor. Security sensors include leak sensor, fire/smoke sensor and surveillance cameras.

Table 8: Types of sensors for smart factory

Sensor Category	**Sensor Types**
Environment Sensors	Temperature sensor
	Light sensor
	Humidity sensor
	Sound sensor
	Vibration sensor
Monitoring sensor	Flow sensor
	Chemical/gas sensor
	Acceleration sensor
Detection sensor	Motion sensor
	Occupancy sensor
Security sensor	Leak sensor
	Fire/smoke sensor
Tracking sensor	Equipment/tags sensor

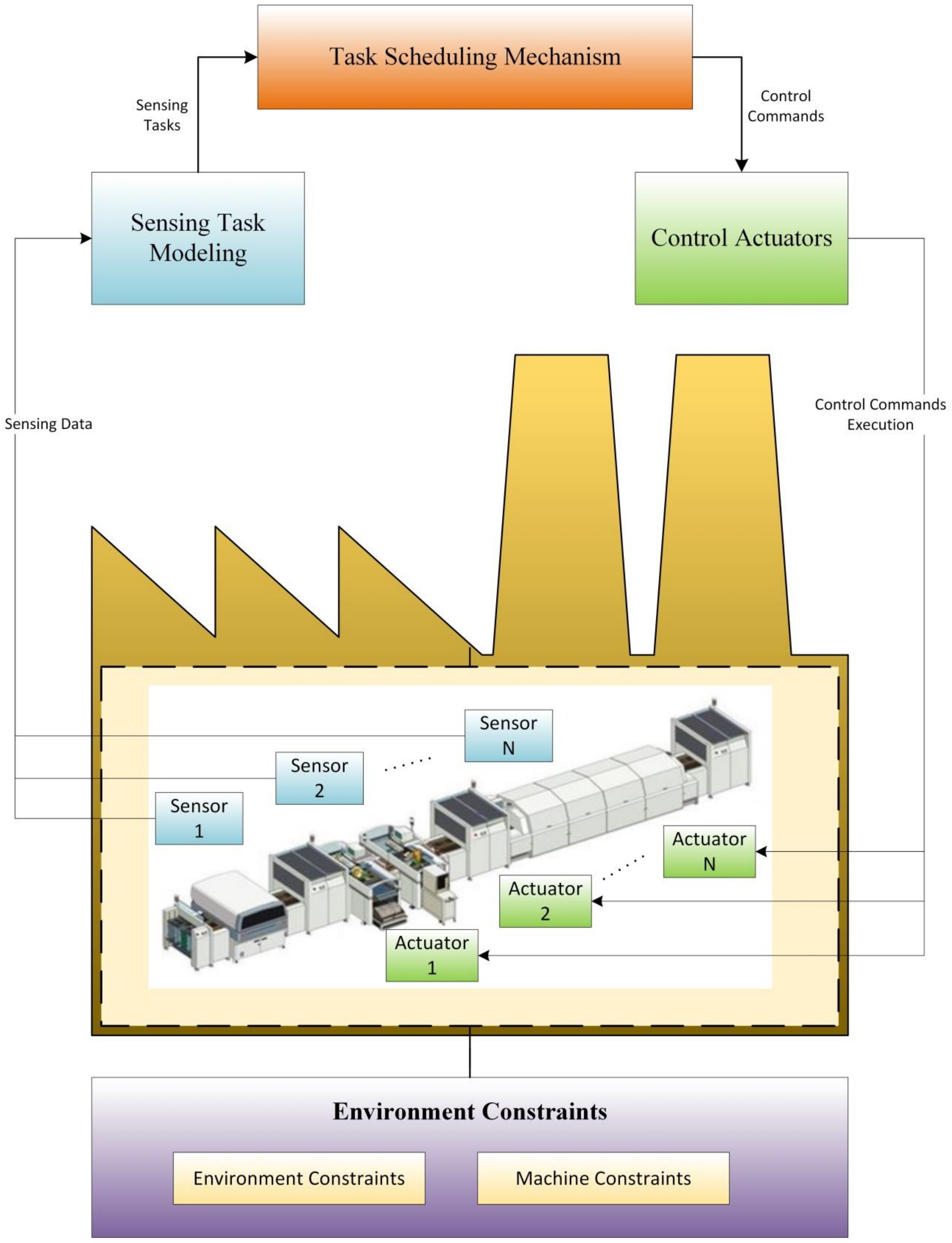


Figure 25: Smart factory environment modeling

The actuators in a smart factory are either the smart factory machines involved in the manufacturing process or the actuators to control the smart manufacturing conditions. The manufacturing machines also vary based on the scenarios. For example, machines in a cake manufacturing might include batter mixing machines, baking oven machine, cake frosting machine, conveyor belt etc. The actuators to maintain the smart manufacturing optimal conditions might include humidifiers, dehumidifier, heater, air conditioning, fans etc.

The Figure 25, presents the smart factory environment modeling where sensors and actuators are installed onto the smart factory. The sensors' readings are transmitted and sensor tasks are modeled and send to scheduling module. The response back to the smart factory is the control of actuators.

The smart factory modeling also has environment constraints and machines constraints which vary from scenario to scenario.

4.2 Input Task Notations

In this section, we design and implement the input tasks model. First, we assume the smart factory model based on number of machines. Let us assume a smart factory with N number of manufacturing machines as shown in equation 12.

$$Smart\ Factory\ Model = \{ M_1, M_2, .., M_N\} \tag{12}$$

$$Constraint\ \{ N > 0\}$$

Each machine will have K number of tasks as shown in equation 13.

$$Machine\ Model = \ \{T_1, T_2, .., T_K\} \tag{13}$$

$$Constraint\ \{ K > 0\}$$

The task parameters in the input task model are TID described as task ID, TA described as task arrival time, TP is described as task priority, JN described as number of jobs required by a task, JID described as jobs ID. The equation 14 explains the task model.

$$Task\ Model = \{\text{TID}, TA, TP, JN, JID_{1,\dots,JN}\} \tag{14}$$

$$Constraint\ \{JN > 0\}$$

Each task can consist of N number of jobs. The jobs parameters are JID described as ID of a job, ST described as start time of a job, ET described as execution time of a job, TB described as time deadline of a job, FT described as proposed finish time of a job, LFT described as latest finish time possible for a job. The equation 15 explains the parameters model for job in a task.

$$Job\ Model = \{\text{JID}, ST, ET, TB, FT, LFT\} \tag{15}$$

$$Constraint\ \{ET > 0\}$$

Job's ID is automatically generated with the generation of a job. The job's ST is time when the job is started at the machine. The execution time of the job is assigned at the time of task and job generation depending on the job type. The job's least finish time is also assigned to the job at time of task and job generation depending on the job type. The job's finish time is computed using the equation 16 and the job's deadline is computed using the equation 17.

$$FT = ST + ET \tag{16}$$

$$TB = LFT - FT \tag{17}$$

4.2.1. Periodic Tasks Set Notation

In the case of periodic tasks, the instances of a periodic task regularly arrive after a set period. Periodic tasks are real-time tasks with a constraint of having the period greater than zero, which means that after a certain amount of time the tasks instance must repeat. Usually, periodic tasks have two states; inactive and runnable. Inactive is the state when the task has not yet arrived at the processor and runnable is the state when the task has arrived again after a certain period and is waiting to run. Equation 18 presents the periodic task model.

$$Periodic\ Task\ Model = \{\text{TID}, TA, TP, P, JN, JID_{1,\dots,JN}\} \tag{18}$$

$$Constraint\ \{P > 0, JN > 0\}$$

Where, TID is the identifier of a periodic task, TA is the arrival time of a periodic task, TP is the priority of the periodic task, P is the period of the periodic task, PB is the preemption bit associated to a periodic task, PB = 1 indicates periodic task can preempt the high priority tasks in case of starvation for a set PT (Preemption Threshold), and PB = 0 indicates periodic task cannot preempt any high priority task.

4.2.2. Event-Driven Tasks Set Notation

An event-driven task is programmed to activate when an event occurs, it can handle any input at any moment. Event-driven tasks have two sub-categories of urgent event-driven tasks and flexible event-driven tasks, these sub-categories help in making the system more flexible. In case of urgent event-driven tasks, tasks should be executed as soon as they arrive at the processor, they cannot wait in the queue. On the other hand, flexible event-driven tasks can afford to wait in the queue but they must also be executed before the deadline. Event-driven tasks have three basic states; inactive, runnable and suspended. Inactive state is when the event to generate the task has not occurred yet, runnable state is when the event is generated and the task is waiting to run, and suspended state is when the event source is triggered off.

An event-driven task with its *i*th execution is denoted as following.

$$EventDriven\ Task\ Model = \{\text{TID}, TA, TP, UB, JN, JID_{1,\dots,JN}\} \tag{19}$$

$$Constraint\ \{\ JN > 0\}$$

Where, TID is the identifier of a periodic task, TA is the arrival time of a periodic task, TP is the priority of the periodic task, *UB* is the urgency bit of an event-driven task, *UB = 1* indicates task is urgent and should be executed ASAP and *UB = 0* indicates non-urgent event-driven tasks.

4.3 Simulation Implementation Environment

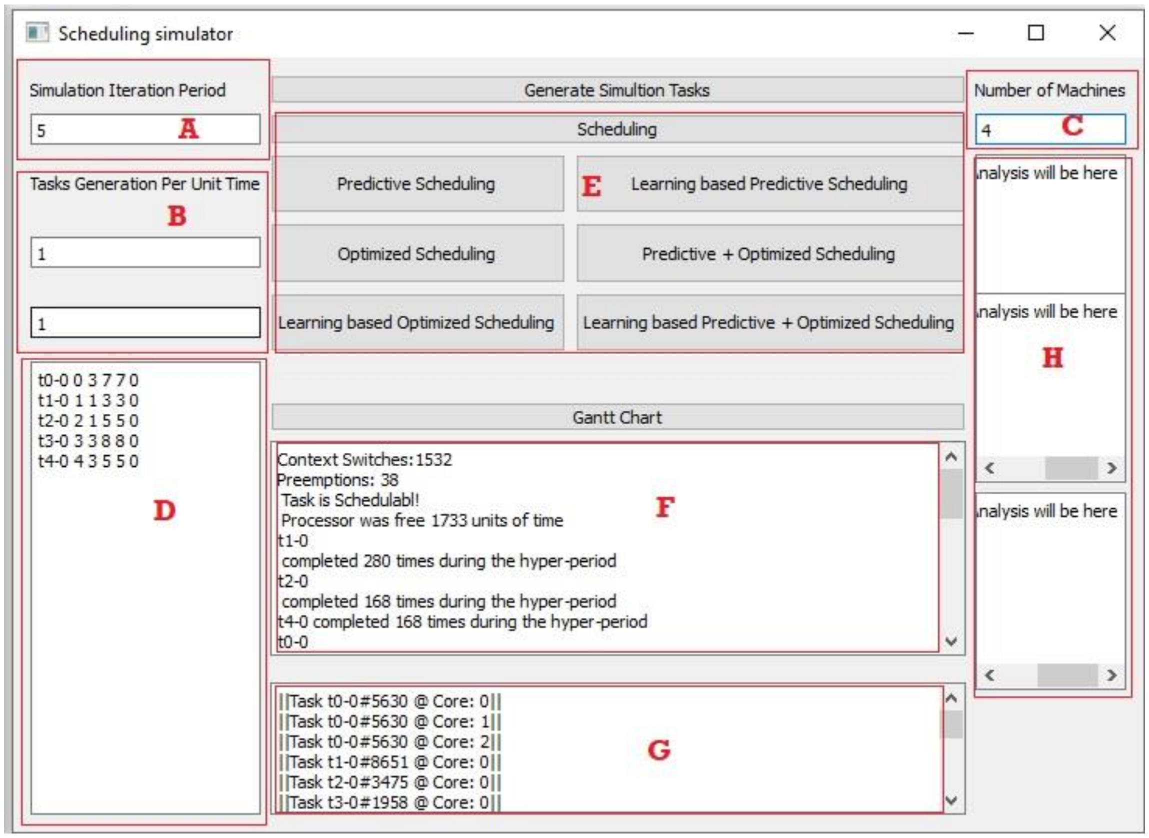
We have used python for implementing the core programming logic of the proposed task management mechanism. Python is a very popular general-purpose programming language; widely used for developing desktop based and web-based applications. We have designed a web-based task simulation visualization tool using PyQt4-based framework. The simulation implementation environment is shown in Table 9.

Table 9: Simulation implementation environment

System Component	**Value**
Operating System	Windows
CPU	Intel ® Core ™ i5-4570 CPU at 3.20 GHz
Primary Memory	8 GB
Programming Language	Python 3.6
User Interface Framework	PyQt4
Machine Core Visualization	HTML

4.4 Scheduling Simulation Application and Visualization

Figure 26 shows main simulation interface window of our implemented smart factory scheduling application. Certain requirements are to be met before starting the scheduling for a given simulation scenario. The inputs of the application are first to be inserted, as highlighted in section A, section B and section C. The section A takes the simulation period as input, section B takes the number of tasks to be generated per simulation unit time that will translate into the arrival rate of the tasks. Section C takes the input of number of total machines at the system. One these inputs are given; the simulation tasks can be generated from the generate simulation tasks button.


Scheduling simulator
Simulation Iteration Period
A
Generate Simultion Tasks
Number of Machines
C
Scheduling
Tasks Generation Per Unit Time
B
Predictive Scheduling
E
Learning based Predictive Scheduling
Optimized Scheduling
Predictive + Optimized Scheduling
Learning based Optimized Scheduling
Learning based Predictive + Optimized Scheduling
nalysis will be here
nalysis will be here
H
nalysis will be here
Gantt Chart
D
F
G

Simulation Iteration Period: 5

Tasks Generation Per Unit Time: 1

1

Number of Machines: 4

```
t0-0 0 3 7 7 0
t1-0 1 1 3 3 0
t2-0 2 1 5 5 0
t3-0 3 3 8 8 0
t4-0 4 3 5 5 0
```

```
Context Switches:1532
Preemptions: 38
 Task is Schedulabl!
 Processor was free 1733 units of time
t1-0
 completed 280 times during the hyper-period
t2-0
 completed 168 times during the hyper-period
t4-0 completed 168 times during the hyper-period
t0-0
```

```
||Task t0-0#5630 @ Core: 0||
||Task t0-0#5630 @ Core: 1||
||Task t0-0#5630 @ Core: 2||
||Task t1-0#8651 @ Core: 0||
||Task t2-0#3475 @ Core: 0||
||Task t3-0#1958 @ Core: 0||
```

Figure 26: Scheduling simulation applications Interface

In section E, the simulation options are available as scheduling which is the baseline scheduling scheme, predictive scheduling (ANN), learning based predictive scheduling (PSO-ANN), optimized scheduling (PSO), learning based optimized scheduling (ANN-PSO), predictive and optimized scheduling, and the last option as learning based predictive and optimized scheduling. The input tasks can be scheduled using any of the scheduling models and output can be visualized.

In section F, the scheduling analysis is provided such as the context switches, number preemptions, tasks set is schedulable or not, machines utilization rate, and number of tasks missed, and number of times tasks are completed during the hyper-period. In section G, the tasks processing order at the machine is listed along with task ID and machine ID. In section H, further analysis comparisons based on predictive scheduling, optimized scheduling and learning based scheduling are to be viewed.

Chapter 5: Simulation and Performance Analysis

In this we present the simulation and performance analysis. In section 5.1, we present the various simulation environments setting for comparison analysis. In section 5.2, we present the simulations and performance analysis for candy box factory tasks dataset. In section 5.3, we present the simulations and performance analysis for simulated tasks dataset. In section 5.4, we present the simulations analysis for machine cluster tasks dataset.

5.1 Simulation Environment for Task Management

In this section, we present the schemes for comparative analysis of the smart factory task management and simulation.

First, we present the *basic FEF scheduling* scheme for the comparative analysis. In our published work in [136], we have presented a fair emergency first (FEF) scheduling scheme. In the paper we have two versions of proposed FEF scheduling algorithm, non-intelligent and intelligent FEF. The intelligent FEF has the addition of learning modules for improving the results. We consider the non-intelligent FEF scheduling algorithm as to be our basic FEF scheme for the comparative analysis.

Next, we compare our proposal with *predictive FEF scheduling* scheme. The FEF scheduling scheme with learning presented in [136] has two learning modules as failure measure (FM) and urgency measure (UM). The learning modules aid the scheduling scheme to improve the tasks completions rate with increase in machine utilization and decreasing the starvation rate. The scheme is only based on the history data predictions and does not involve any optimization mechanism.

Next, we add the PSO based learning to prediction with hybrid ACM-FEF scheduling algorithm and refer to it as *learned predictive hybrid scheduling* scheme.

At final stage is our proposal, when we add the learning to optimization module to the learning to prediction based hybrid ACM-FEF scheduling scheme and refer to it as *learned predictive and optimized hybrid scheduling* scheme.

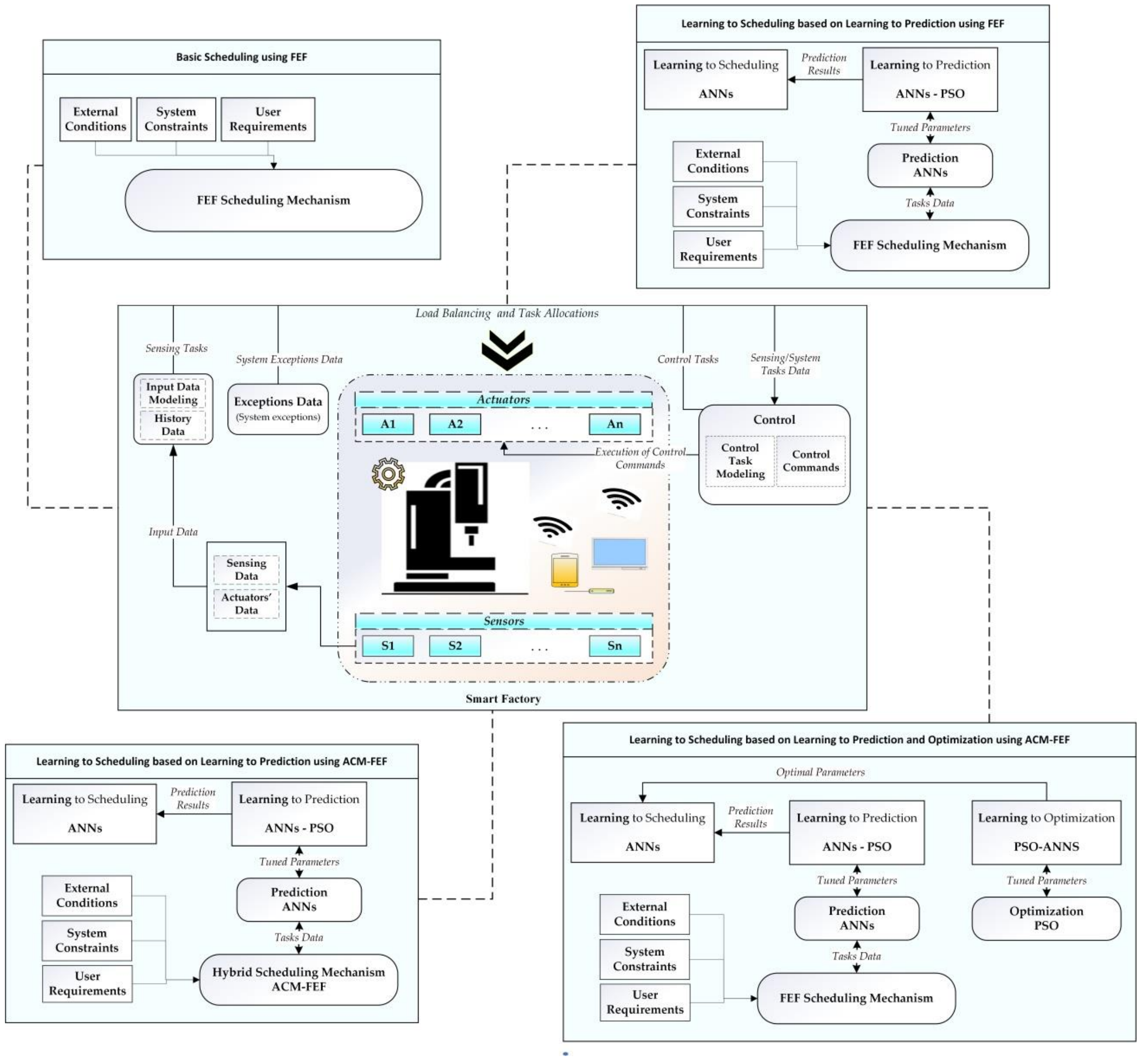


Figure 27: Scheduling schemes for simulations of tasks management in smart factory

5.2 Simulations and Performance Analysis of Candy Box Factory

In this section we build a scenario for smart factory modeling. We have assumed a use case scenario of candy box assembling and packaging in a smart factory. We have built the candy box use case scenario based on the candy packaging application case study presented in [139]. To our assumption, the candy smart factory manufactures N types of candies, and makes custom candy combination boxes based on the user orders.

In this use case, we do not consider the manufacturing process of candies. We consider a scenario where the N types of candies have been manufactured and the task now is to get the customized orders from the users, assemble customized candy boxes based on the user orders and forward them to pack the boxes to be ready to be delivered.

In our built scenario we have three sensors, four actuators and eight smart manufacturing machines. The sensors are of two types as ambient sensors and on-machine sensors (Figure 28). The ambient sensors are temperature sensor and humidity sensor. The on-machine sensor is occupancy sensor. Table 10 shows the involved sensors in the candy production scenario. The actuators involved are heater, chiller, humidifier, and dehumidifier. Table 11 shows the operation levels of actuators involved in the candy production.

Table 10: Sensors list for candy box factory

Sensors Type	Sensor Name	Value Range
Ambient sensors	Temperature sensor	21- 24 °C [140]
	Humidity sensor	40% - 35% [140]

On-machine sensors	Occupancy sensor	0/1

Table 11: Actuators list for candy box factory

Actuator Type	Actuator Name	Controls
Heater	Assembly Machines	Busy/Idle
	Packaging Machines	Busy/Idle
Chiller Humidifier	Chiller	On/Off/Adjust Level
	Heater	On/Off/Adjust Level
	Humidifier	On/Off/Adjust Level
	Dehumidifier	On/Off/Adjust Level

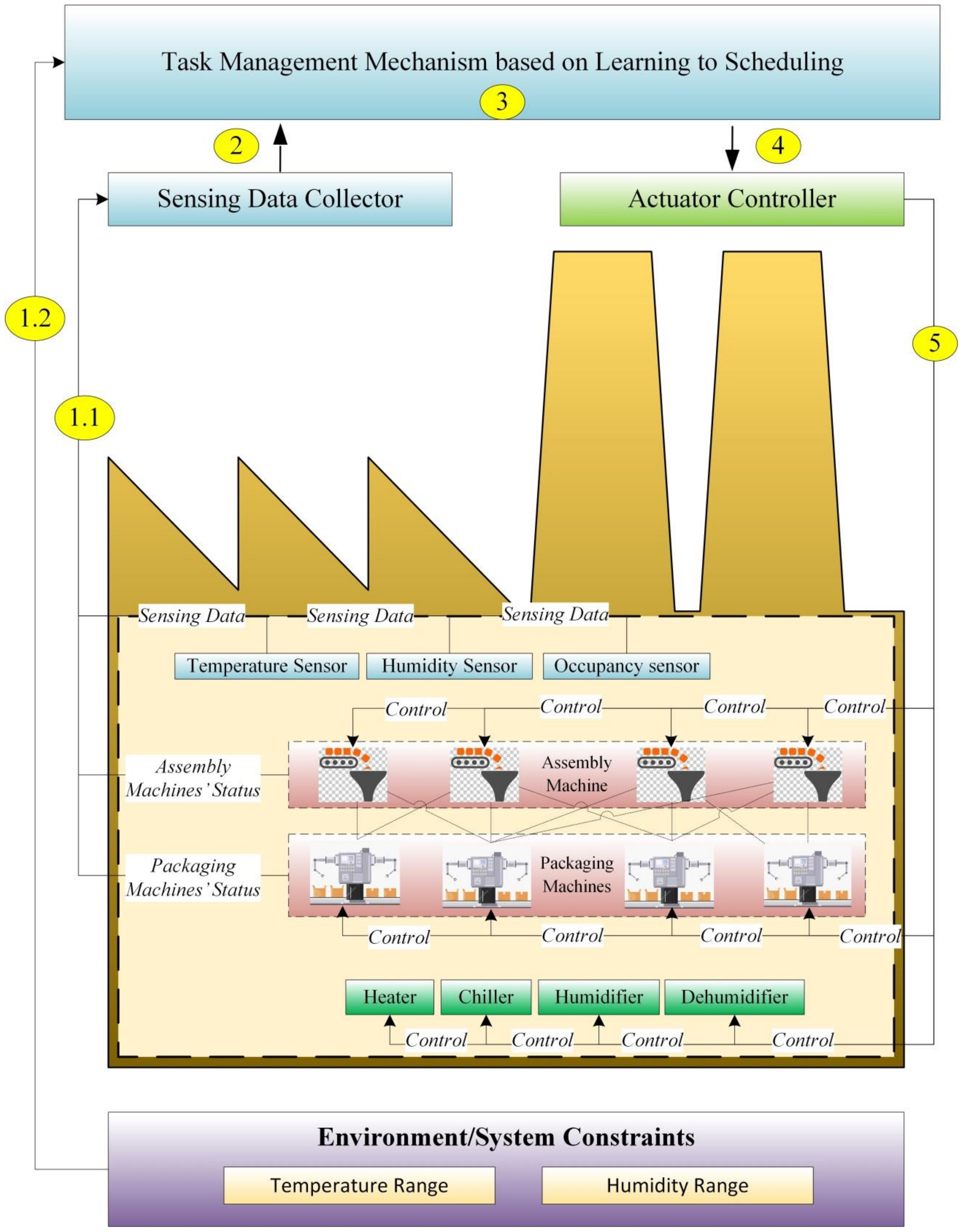


Figure 28: Scenario modeling for candy box factory

Figure 29 presents the overall simulation flow for the tasks generation, sensing data based control tasks generation and system data based control tasks generation.

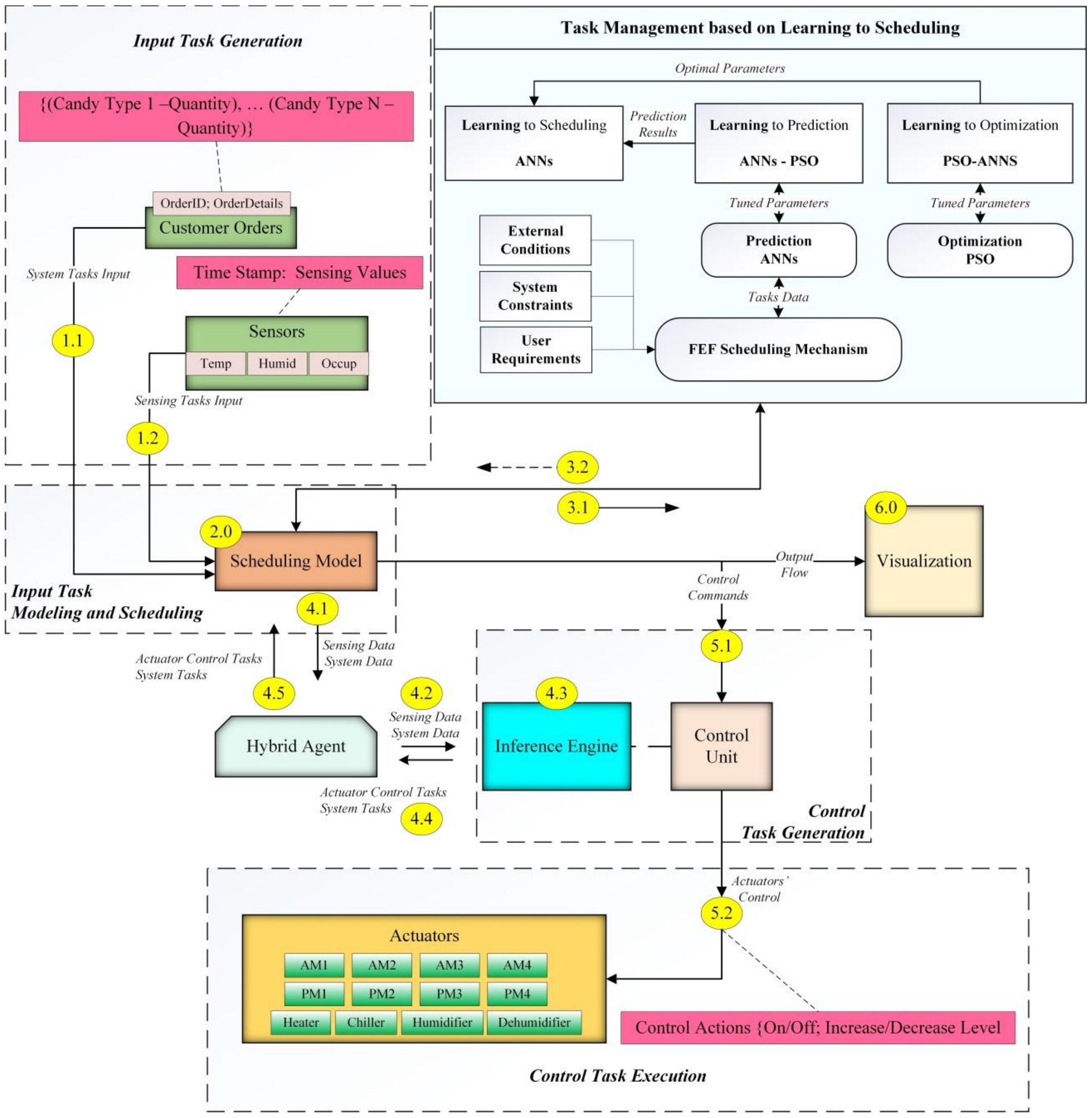


Figure 29: Overall scheduling simulation flow for candy box factory

The sensing tasks from sensor reading are modeled and passed onto the scheduling module and the system tasks from customers' orders of candy box are modeled and passed onto the scheduling module. The scheduling policies refer to the optimization and prediction based scheduling mechanism and combinations. The tasks are scheduled and upon execution the sensing values or system values are passed onto the inference engine where all the rules for the system conditions are listed. The inference engine matches the values and generates the control tasks in response which are sent back onto the scheduler and upon execution the control commands are sent to the actuators.

We have eight machines in total which represent the two phases of candy assembly and packaging scenario. First phase is to assemble the customized candy box based on the user order. Second phase is to pack the customized candy box to get ready to be delivered. In first phase we have assembly machines and in second phase we have packaging machines. Table 12 shows the machines details.

Table 12: Machines' attributes for candy box factory

Machine Type	**No. of Installed Machines**	**Input**	**Output**
Assembly Machines [AM]	4 {AM1, AM2, AM3, AM4}	[{Order Number}, {AM-ID}, {(Candy Type 1 – Quantity), … (Candy Type N –Quantity)}]	[{Order Number}, {PM-ID}, {Package Style}]]
Packaging Machines [PM]	4 {PM1, PM2, PM3, PM4}	[{Order Number}, {PM-ID}, {Package Style}]]	[{Order Number}, {Delivery ID}]

5.2.1 Input Tasks Modeling of Candy Box Factory

In this sub-section we define the task generation phase for the candy box use case based on our task design and modeling presented in the section 4.2.

We have three types of tasks as sensing tasks, system tasks and the control tasks. The sensing tasks are generated based on the sensor value readings and are periodic in nature. The period of sensing tasks depends on the time interval after which the sensing values from the sensors are gathered. The sensing tasks when executed will trigger the control tasks based on the conditions. The system tasks include the operational movements of the manufacturing machines which are based on the customer orders. The system tasks when executed will trigger the control tasks. The control tasks are modeled from the responses triggered from sensing tasks and system tasks. The execution of control tasks sends the control commands to the actuators, installed onto the smart factory, in order to assemble the candy box and pack the candy box.

The sensing tasks will include the parameters as task ID, task arrival time, task priority, period and pre-emption bit.

Table 13: Sensing tasks parameters for candy box factory

Sensing Task	**Task Type**	**Task Parameters**
Temperature Sensing Tasks	Periodic	{TID, TA, TP, P, PB}
Humidity Sensing Tasks	Periodic	{TID, TA, TP, P, PB}
Occupancy Sensing Tasks	Periodic	{TID, TA, TP, P, PB}

A customer will visit the factory site; select the desired candy combinations to place a customized candy box order. The order is then forwarded to the smart factory's production unit where the order is processed for assembling and packing the candy box to be delivered.

The system tasks are generated based on the customers' orders of customized candy box. The system tasks parameters will include the parameters as task ID, task arrival time, task execution time, task deadline, task proposed finish time, task least finish time, task priority, and urgency bit. Table 14 shows the system tasks' details.

Table 14: System tasks parameters for candy box factory

System Task	**Task Type**	**Task Modeling**	**Order Details**
Candy Box Order Placement	Event Driven	{TID, TA, ET, TB, FT, LFT, TP, UB}	{(Candy Type 1 –Quantity), … (Candy Type N – Quantity)}

The sensing tasks and system tasks trigger events which result into the modeling of control tasks. The control tasks are used to control the actuators onto the smart factory environment. In this scenario, we have a total of twelve actuators including eight manufacturing machines, one chiller, one heater, one humidifier and one dehumidifier system.

The control tasks for chiller, heater, humidifier and dehumidifier are generated in response to the sensing tasks of temperature and humidity. If sensed temperature values are greater than maximum temperature then temperature is tuned by increasing chiller system levels. If sensed temperature values are less than minimum values then temperature is tuned by increasing heater system levels. Similarly, if sensed humidity values are greater than maximum humidity then humidity is tuned by increasing dehumidifier system levels; and if sensed humidity values are less than minimum values then humidity is tuned by increasing humidifier system levels (Table 15).

Table 15: Control task triggered from sensing task for candy box factory

Sensing Value	Control Action	Controls	Output Status
Temp > 24 °C	Chiller On	Increase Chilling Level	1
	Heater Off	Decrease Heating Level	0
Temp < 21 °C	Chiller Off	Decrease Chilling Level	0
	Heater On	Increase Heating Level	1
Humidity < 35%	Humidifier On	Increase Humidifier Level	1
	Dehumidifier Off	Decrease Dehumidifier Level	0
Humidity > 40%	Humidifier Off	Decrease Humidifier Level	0
	Dehumidifier On	Increase Dehumidifier Level	1

The control tasks for eight manufacturing machines are generated in response to the system task of candy box order placement. The response event driven task is candy box order response task with two jobs as shown in Table 16.

Table 16: Control task triggered from system task for candy box factory

System Task	Task Type	Number of Jobs	Job 1	Job 2
Candy Box Order Response	Event Driven	2	Assembly Candy Box	Pack Candy Box

The candy box order response task has two jobs; job one is to assembly the candy box and job 2 is to pack the candy box. The input job modeling based on section 4.2 is given in the Table 17 below.

Table 17: Control task's jobs list for candy box factory

List of Jobs	Input Job Modeling
Job 1: Assembly Candy Box	Job Model = {J-1, ST, ET, TB, FT, LFT}
Job 2: Pack Candy Box	Job Model = {J-2, ST, ET, TB, FT, LFT}

Table 18 shows the job execution details. The job 1 can be executed on any of the four manufacturing machines AM1, AM2, AM3, and AM4. The job 2 can be executed on any of the manufacturing machines from PM1, PM2, PM3, and PM4. The machine selection will be done using the scheduling algorithm, based on each machine's status and load.

Table 18: Control task's jobs description for candy box factory

List of Jobs	Available Machines	Job Input + Order Input	Output Status
J-1 : Assembly Candy Box	{AM1, AM2, AM3, AM4}	[{J-1}, {ST}, {ET}, {TB}, {FT}, {LFT},{Order Number}, {AM-ID}, {(Candy Type 1 –Quantity), … (Candy Type N –Quantity)}]	[{Order Number}, {PM-ID}, {Package Style}]]
J-2 : Pack Candy Box	{PM1, PM2, PM3, PM4}	[{J-1}, {ST}, {ET}, {TB}, {FT}, {LFT}, {Order Number}, {PM-ID}, {Package Style}]]	[{Order Number}, {Delivery ID}]

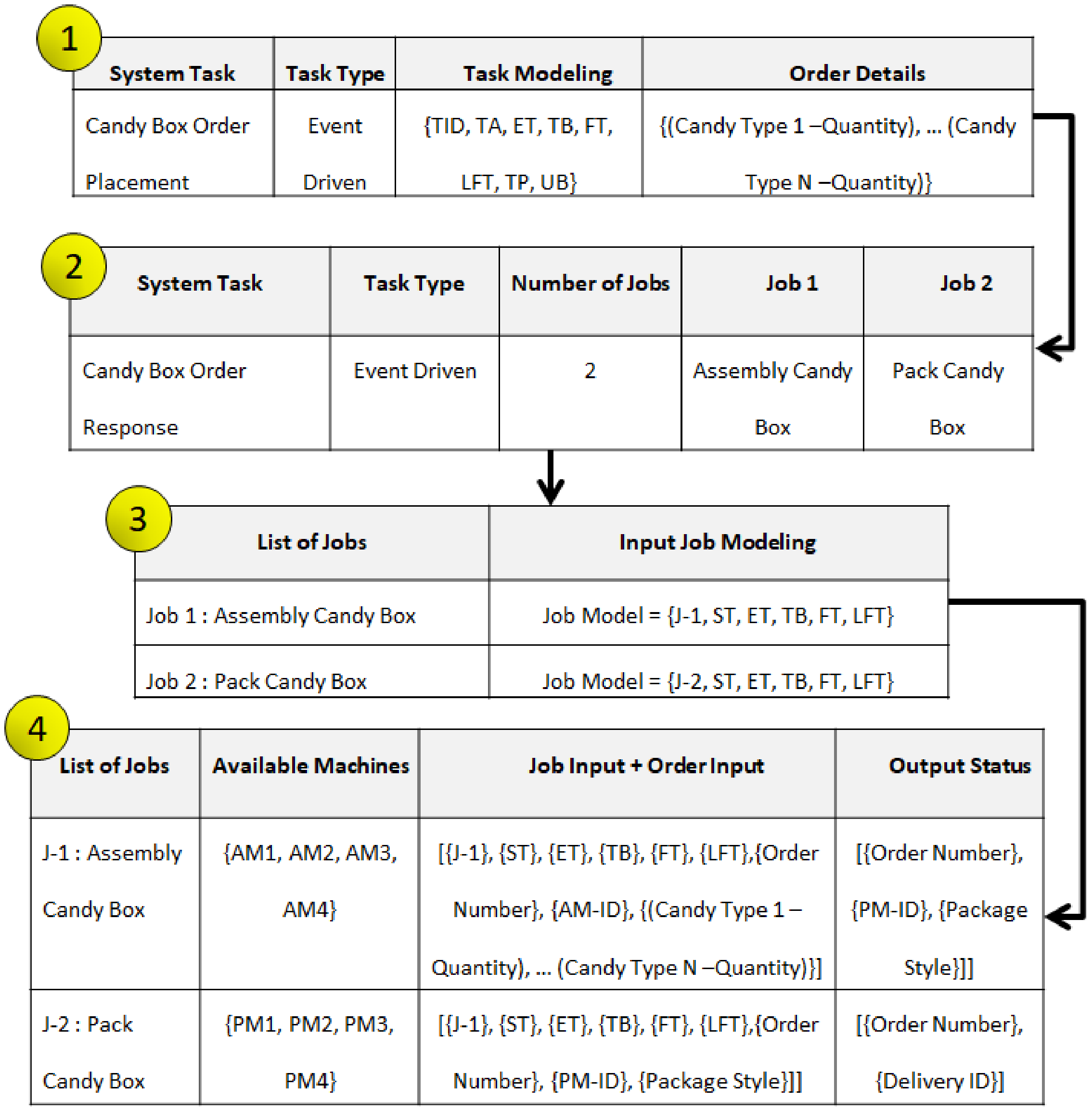
5.2.2 Tasks Simulation and Performance Analysis for Candy Box Factory

In this sub-section we present the tasks simulation and performance analysis for candy box factory use case scenario.

5.1.2.1 Event Driven Tasks Simulation Flow

In this sub-section, we present the simulation flow for the event-driven tasks generation in the given scenario of candy box factory.

The tasks flow is initiated with an event driven task of candy order placement (Figure 30; Point 1). The order placement task is a system task generated when users' place candy box order. Once the order placement task is triggered, next the system generates candy box order response task (Figure 30; Point 2). Candy box order response task is also system task which has two jobs as assembly candy box (Job 1) and pack candy box (Job 2). The input job modeling is based on the task modeling introduced in section 4.2. (Figure 30, Point 3). Figure 30 (Point 4), shows the job execution details. Where, J-1 can be executed on any of the four manufacturing machines AM1, AM2, AM3, and AM4; and J-2 can be executed on any of the manufacturing machines from PM1, PM2, PM3, and PM4. The machine selection, for executing J-1 in the assembly machine network and executing J-2 in the packaging machine network, will be done based on each machines' current status and load.



System Task	Task Type	Task Modeling	Order Details
Candy Box Order Placement	Event Driven	{TID, TA, ET, TB, FT, LFT, TP, UB}	{(Candy Type 1 –Quantity), … (Candy Type N –Quantity)}

System Task	Task Type	Number of Jobs	Job 1	Job 2
Candy Box Order Response	Event Driven	2	Assembly Candy Box	Pack Candy Box

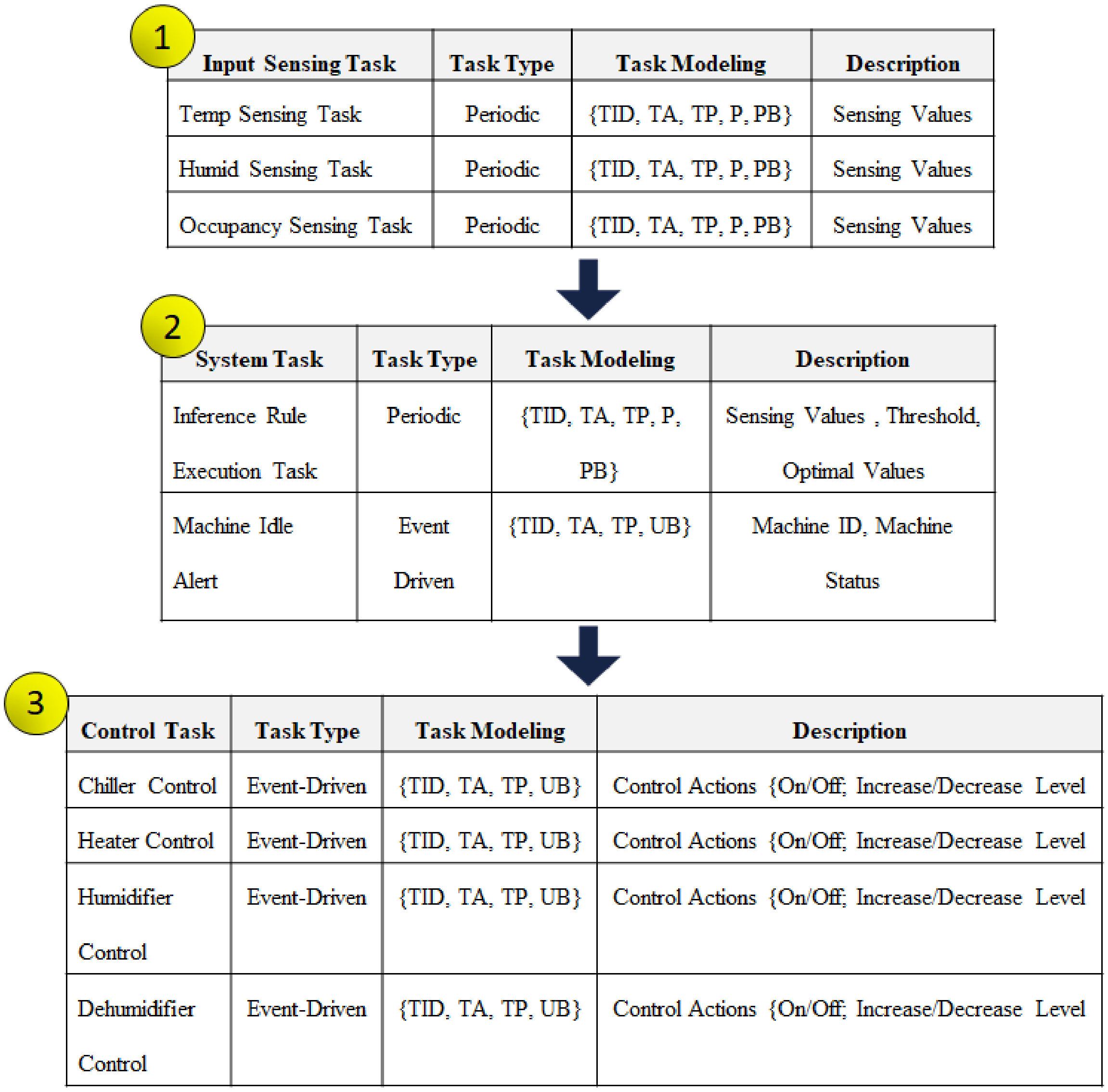
List of Jobs	Input Job Modeling
Job 1 : Assembly Candy Box	Job Model = {J-1, ST, ET, TB, FT, LFT}
Job 2 : Pack Candy Box	Job Model = {J-2, ST, ET, TB, FT, LFT}

List of Jobs	Available Machines	Job Input + Order Input	Output Status
J-1 : Assembly Candy Box	{AM1, AM2, AM3, AM4}	[{J-1}, {ST}, {ET}, {TB}, {FT}, {LFT},{Order Number}, {AM-ID}, {(Candy Type 1 –Quantity), … (Candy Type N –Quantity)}]	[{Order Number}, {PM-ID}, {Package Style}]]
J-2 : Pack Candy Box	{PM1, PM2, PM3, PM4}	[{J-1}, {ST}, {ET}, {TB}, {FT}, {LFT},{Order Number}, {PM-ID}, {Package Style}]]	[{Order Number}, {Delivery ID}]

Figure 30: Event driven tasks simulation flow for candy box factory

5.1.2.2 Periodic Tasks Simulation Flow

In this sub-section, we present the simulation flow for the periodic tasks' generation in the given scenario of candy box factory.



1

Input Sensing Task	Task Type	Task Modeling	Description
Temp Sensing Task	Periodic	{TID, TA, TP, P, PB}	Sensing Values
Humid Sensing Task	Periodic	{TID, TA, TP, P, PB}	Sensing Values
Occupancy Sensing Task	Periodic	{TID, TA, TP, P, PB}	Sensing Values

2

System Task	Task Type	Task Modeling	Description
Inference Rule Execution Task	Periodic	{TID, TA, TP, P, PB}	Sensing Values , Threshold, Optimal Values
Machine Idle Alert	Event Driven	{TID, TA, TP, UB}	Machine ID, Machine Status

3

Control Task	Task Type	Task Modeling	Description
Chiller Control	Event-Driven	{TID, TA, TP, UB}	Control Actions {On/Off; Increase/Decrease Level
Heater Control	Event-Driven	{TID, TA, TP, UB}	Control Actions {On/Off; Increase/Decrease Level
Humidifier Control	Event-Driven	{TID, TA, TP, UB}	Control Actions {On/Off; Increase/Decrease Level
Dehumidifier Control	Event-Driven	{TID, TA, TP, UB}	Control Actions {On/Off; Increase/Decrease Level

Figure 31: Periodic tasks simulation flow for candy box factory

The periodic tasks flow initiates three sensing tasks as temperature sensing task, humidity sensing task and occupancy sensing task (Figure 31; Point 1). The sensing task's period defines the set interval, after which sensing data is to be collected from sensors installed in the candy box factory. Next, in response to sensing data values of humidity, temperature and occupancy inference rule execution task is generated. Inference rule execution task generates when given data values map to set thresholds and controls or alerts are to be triggered in response. In response to machine occupancy inference rule execution task, next task generated is machine idle alert (Figure 31, Point 2). The control tasks executed in response to temperature values inference rule execution task are chiller control and heater control. The control tasks executed in response to humidity values inference rule execution task are humidifier control and dehumidifier control (Figure 31, Point 3).

5.1.2.3 Simulations for Candy Box Factory

The execution time for sensing tasks is set to be 20 milliseconds (ms) and the priority is set to be normal periodic tasks. The execution time for system tasks is set to be 300 milliseconds (ms) and the priority for order placement task is set to be urgent event driven task and the priority for the inference rule execution task is set to be priority periodic task. The execution times for all the control tasks are set to be 520 milliseconds (ms) and the priorities for environmental conditions control actuator (heater, chiller, humidifier, and dehumidifier) are set to be urgent event driven. The priorities for control tasks of manufacturing machines (AM1, AM2, AM3, AM4, PM1, PM2, PM3, and PM4) are set based on the priority set at customers' order time and deadline. It can be either normal event driven or urgent event driven task.

Table 19: List of tasks execution times and priority type for candy box factory

Task	**CPU Time Required**	**Task Priority**
Sensing Tasks		
Temperature Sensing Task	20ms	Normal Periodic Task

Humidity Sensing Task	20ms	Normal Periodic Task
Occupancy Sensing Task	20ms	Normal Periodic Task
System Tasks		
Order Placement System Task	300ms	Urgent Event Driven Task
Inference Rule Execution Task	300ms	Normal Periodic Task
Control Tasks		
Heater Control Task	520ms	Urgent Event Driven Task
Chiller Control Task	520ms	Urgent Event Driven Task
Humidifier Control Task	520ms	Urgent Event Driven Task
Dehumidifier Control Task	520ms	Urgent Event Driven Task
Control AM1	520ms	Normal/ Urgent Event Driven Task
Control AM2	520ms	Normal/ Urgent Event Driven Task
Control AM3	520ms	Normal/ Urgent Event Driven Task
Control AM4	520ms	Normal/ Urgent Event

		Driven Task
Control PM1	520ms	Normal/ Urgent Event Driven Task
Control PM2	520ms	Normal/ Urgent Event Driven Task
Control PM3	520ms	Normal/ Urgent Event Driven Task
Control PM4	520ms	Normal/ Urgent Event Driven Task

Our first step is data collection and data generation. Our required data includes the collection of history data for training in learning processes. The history data include tasks' data as task arrival time, execution time, deadline, finish time, time budget, allocated machine, allocated machines' load, and capacity requirements of the machine. The tasks completion status and machine utilization history data are also collected in order to train the prediction module for future predictions. The history data is collected by simulation iterations of tasks generations and simulation iterations. In Figure 32, we present 30 instances of the history tasks' data. The history tasks' data is collected by simulation iterations of the tasks. In Figure 33, we present the history data collection sample for tasks completion status and in Figure 34, we show the history data collection sample for machine utilization rate.

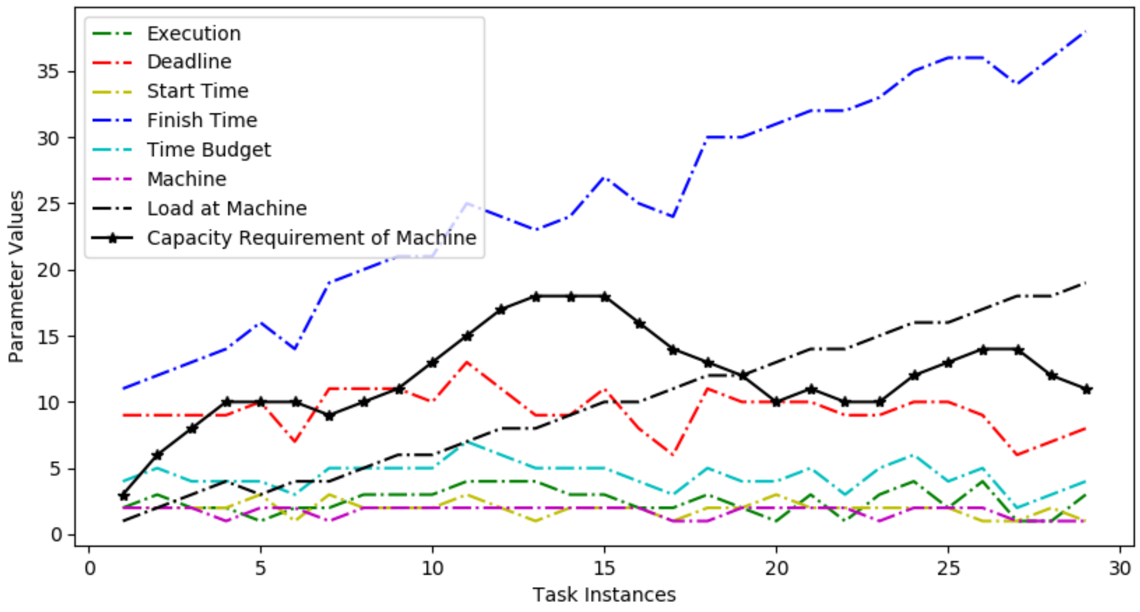


Figure 32: Tasks simulation training data for candy box factory

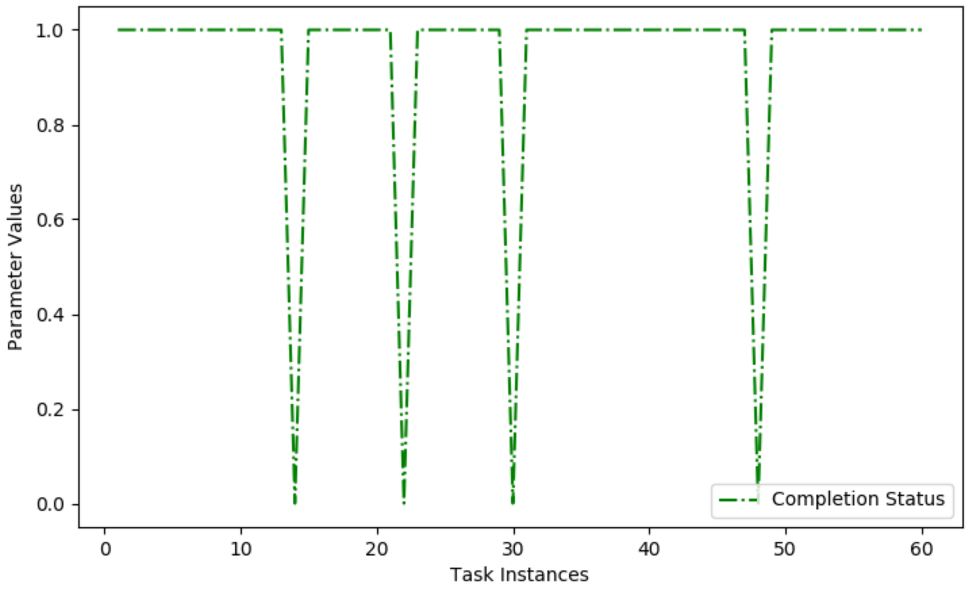


Figure 33: Tasks completion status training data for candy box factory

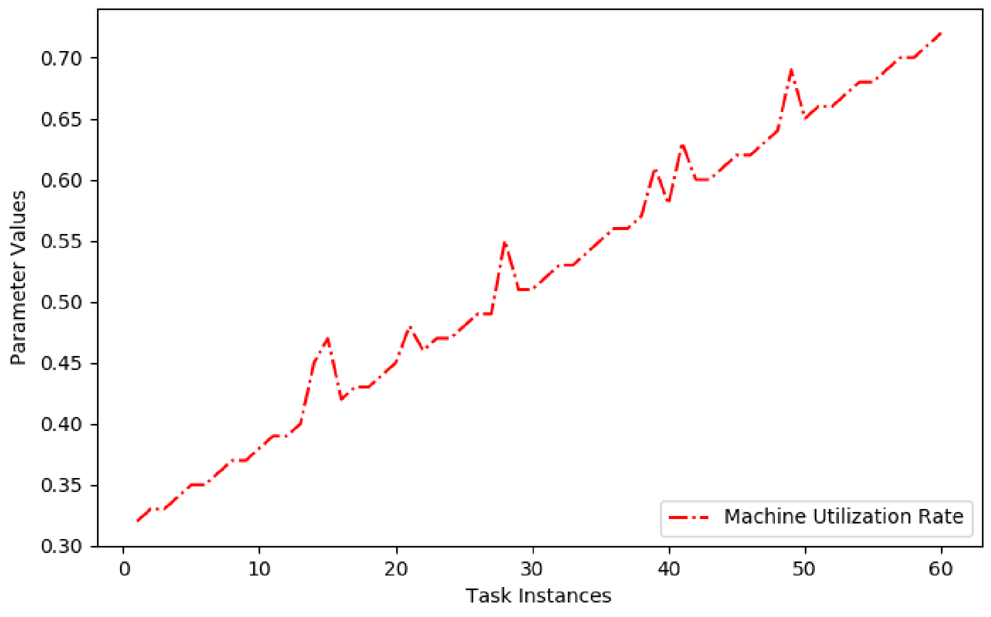


Figure 34: Machine utilization training data for candy box factory

Next, we simulated the temperature and humidity data for the training purposes. The simulated temperature data varies mostly between the valid temperature ranges with randomly inserted out of range fluctuations so that rule engine can detect and control the anomaly sensing values for the temperature. Similarly, for the humidity we have followed the same mechanism of simulating the data between the valid ranges with some randomly inserted out of range fluctuations so that rule engine can detect and control the anomaly sensing values for the humidity.

In Figure 35, we present the simulated data sample instances for temperature sensing data and the simulated data sample instances for humidity sensing data.

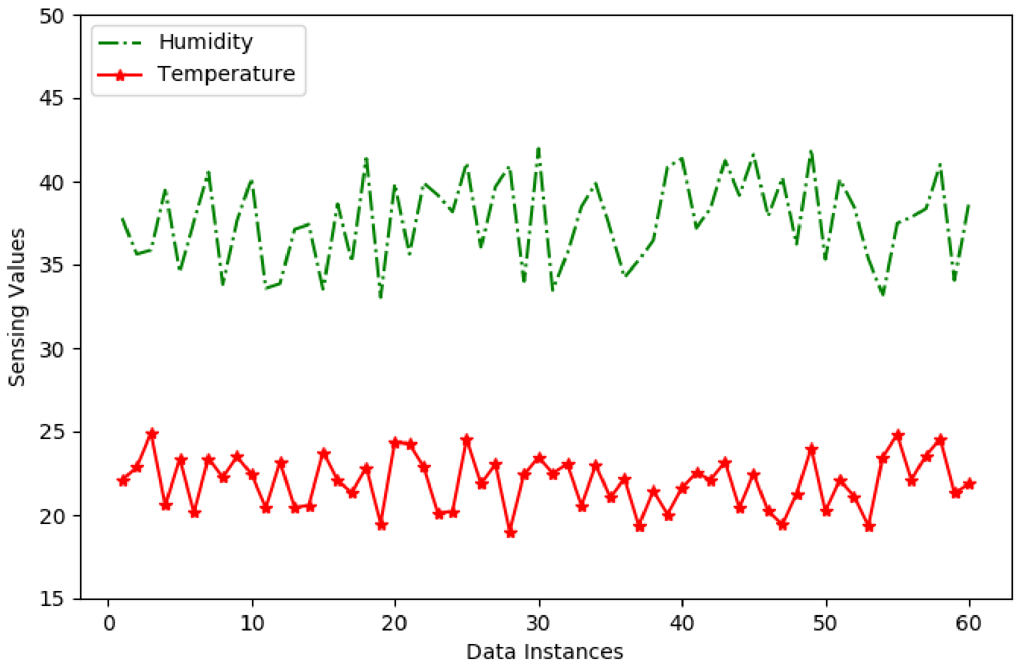
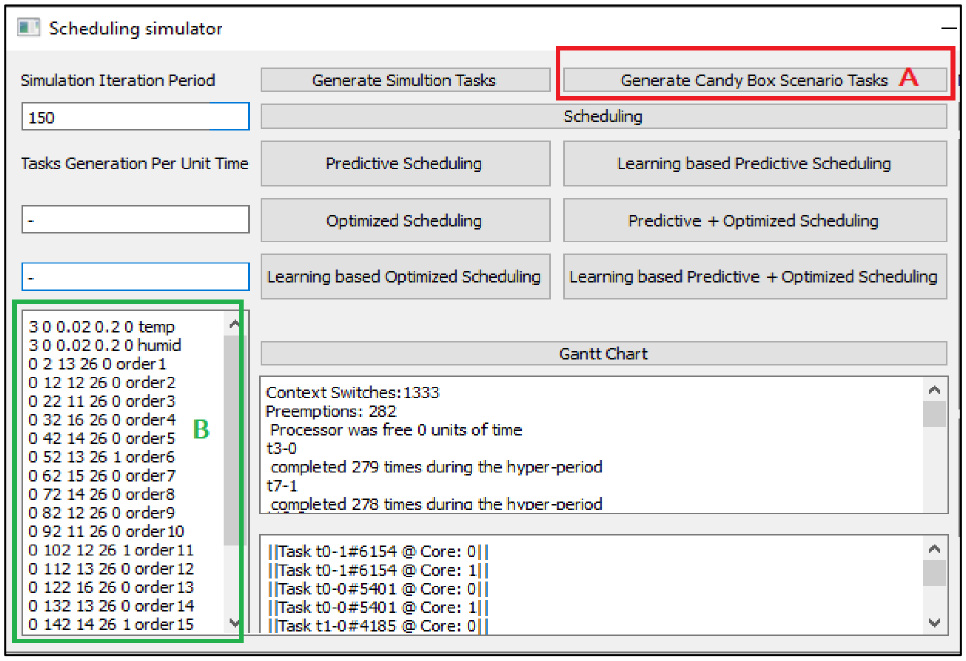


Figure 35: Simulated sensing data for temperature and humidity for candy box factory



Scheduling simulator
Simulation Iteration Period
Generate Simultion Tasks
Generate Candy Box Scenario Tasks
A
Scheduling
Tasks Generation Per Unit Time
Predictive Scheduling
Learning based Predictive Scheduling
Optimized Scheduling
Predictive + Optimized Scheduling
Learning based Optimized Scheduling
Learning based Predictive + Optimized Scheduling
B
Gantt Chart

```
150
-
-
```

```
3 0 0.02 0.2 0 temp
3 0 0.02 0.2 0 humid
0 2 13 26 0 order1
0 12 12 26 0 order2
0 22 11 26 0 order3
0 32 16 26 0 order4
0 42 14 26 0 order5
0 52 13 26 1 order6
0 62 15 26 0 order7
0 72 14 26 0 order8
0 82 12 26 0 order9
0 92 11 26 0 order10
0 102 12 26 1 order11
0 112 13 26 0 order12
0 122 16 26 0 order13
0 132 13 26 0 order14
0 142 14 26 1 order15
```

```
Context Switches:1333
Preemptions: 282
 Processor was free 0 units of time
t3-0
 completed 279 times during the hyper-period
t7-1
 completed 278 times during the hyper-period
```

```
||Task t0-1#6154 @ Core: 0||
||Task t0-1#6154 @ Core: 1||
||Task t0-0#5401 @ Core: 0||
||Task t0-0#5401 @ Core: 1||
||Task t1-0#4185 @ Core: 0||
```

Figure 36: Tasks generation and simulation for candy box factory

Figure 36 shows the candy box use case task generation and simulation output. We have added a scenario simulation based task generation function highlighted in red color as section A. The tasks are generated based on the scenario details as mentioned above and then scheduled based on the selected scheduling mechanism. Figure 37 shows the tasks instances visualization while executing at machines.

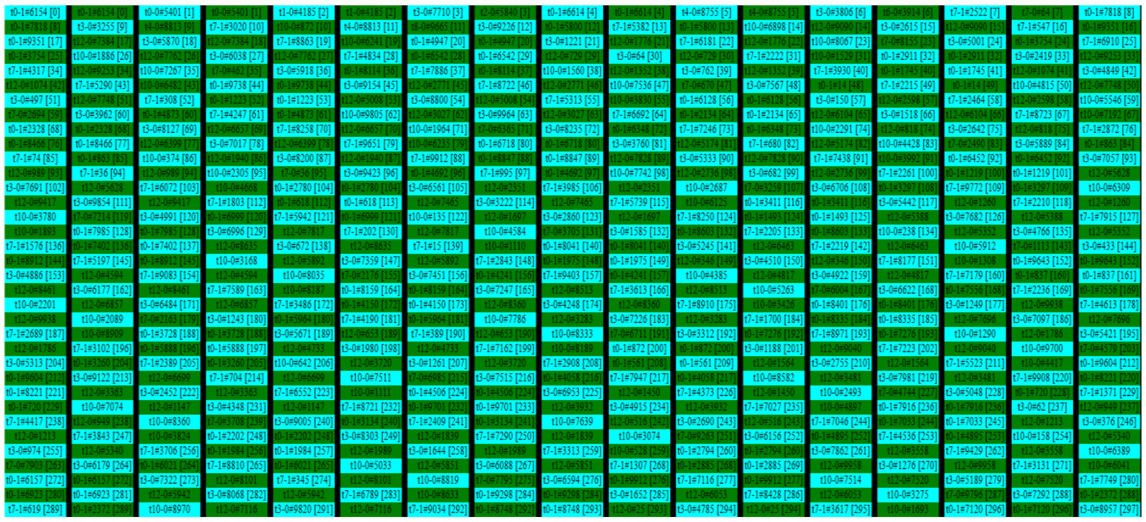


Figure 37: Tasks Simulation flow on machine for candy box factory

Figure 38 presents the prediction accuracy comparisons using ANN prediction algorithm and PSO based ANN (PSO-ANN) prediction algorithm. The results show a significant improve in terms of minimized number of epochs and increased prediction accuracy. The maximum prediction accuracy achieved using ANN is 99.02% in 600 iterations and the maximum prediction accuracy achieved using PSO-NN is 99.39% in 700 iterations. At 600 iterations, the PSO-NN gets to the prediction accuracy of 99.27% which is still greater than ANN at 600 iterations.

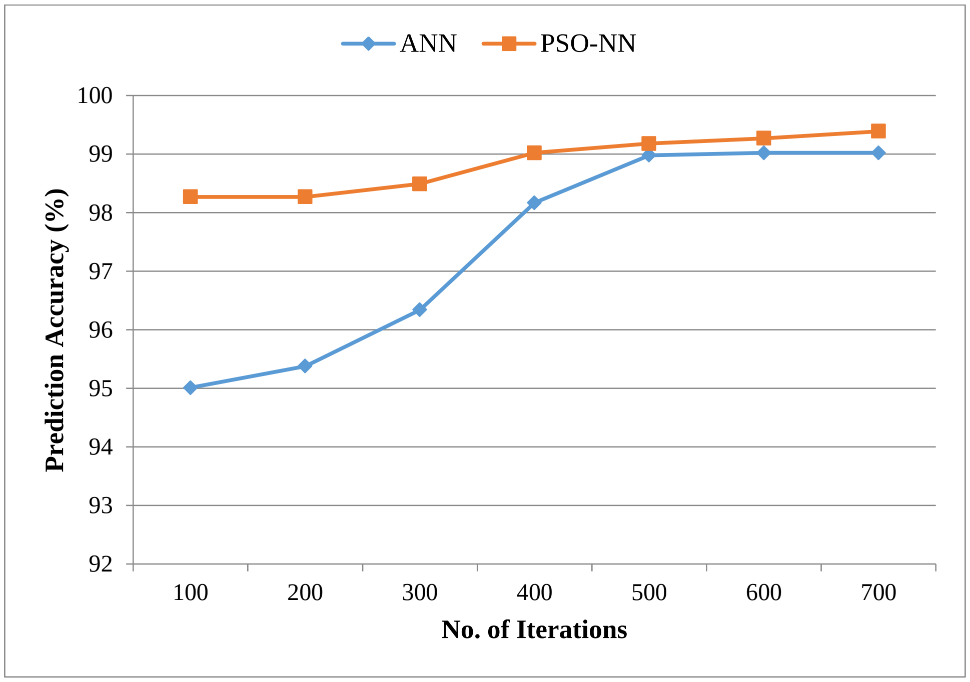


Figure 38: Prediction accuracy comparisons for ANN and PSO-NN for candy box factory

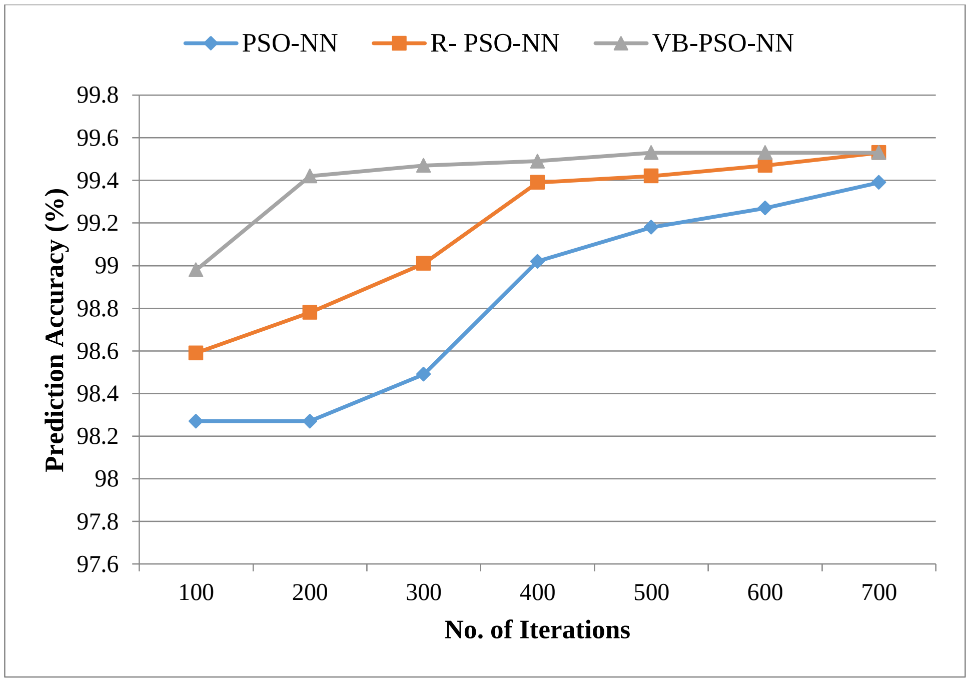
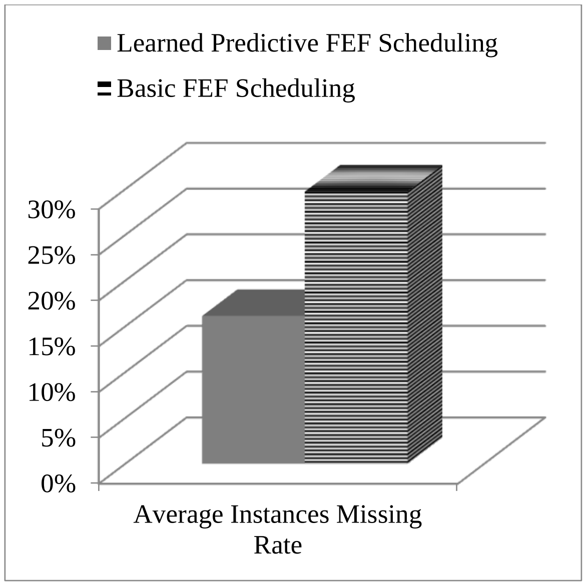
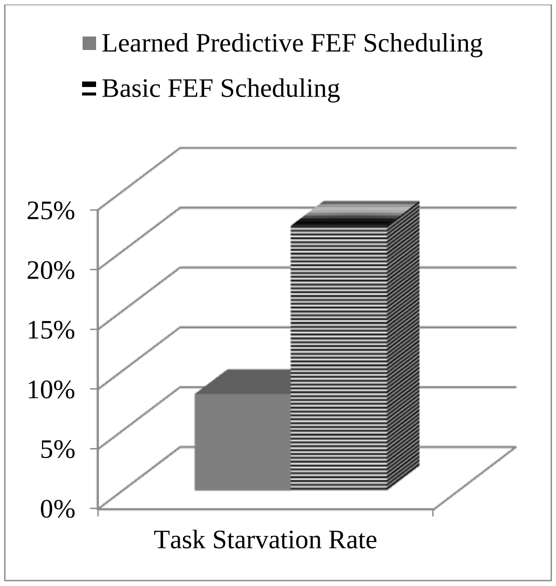


Figure 39: Prediction accuracy comparisons for R-PSO-NN and VB-PSO-NN for candy box factory

In Figure 39, we present the comparisons of prediction accuracy based on PSO based ANN along with the variations of PSO based ANN as R-PSO-NN and VB-PSO-NN. The results show the prediction accuracy of R-PSO-NN and VB-PSO-NN is further improved from PSO-NN. In candy box factory predictions, VB-PSO-NN gives maximum prediction accuracy with least number of iterations. The VB-PSO-NN performance is followed by R-PSO-NN, which gives slightly less prediction accuracy. The maximum prediction accuracy achieved using PSO-NN is 99.39% in 700 iterations while the maximum prediction accuracy achieved using R-PSO-NN is 99.53% in 700 iterations and the maximum prediction accuracy achieved using VB-PSO-NN is 99.53% in 500 iterations.





(a) **(b)**

Figure 40: Proposed scheduling comparisons with prediction and without prediction for candy box factory (a) average instances missing rate in percentage; (b) average task starvation rate in percentage

In Figure 40, we present the comparisons of basic FEF with the learned prediction FEF with an aim to demonstrate the effect of learning to prediction in the scheduling algorithm. The comparisons are performed on the candy box factory data. The results show the percentage of task starvation rate and average instances missing rate in the simulations. It can be clearly observed that learned predictive FEF has a smaller number of starved tasks rate and a smaller number of tasks instances missing rate. The FEF scheduling algorithm has around 22.12 % of tasks starved and 29.75 % of tasks instances missed whereas the learned predictive FEF scheduling has around 8 % of tasks starved and 16 % of tasks instances missed. The learning of prediction module in learning to scheduling mechanism increases the overall performance of the scheduler as it enables the scheduler to make informed and learned decisions.

In the Figure 41, we present the comparisons among predictive FEF scheduling and learned predictive FEF scheduling. The graph shows the tasks average response times at y-axis and test iterations at x-axis. A significant decrease in the tasks' response time is observed with the addition of learning of module. The average response time for tasks set using predictive FEF scheduling is 2191.39 milliseconds and the average response time for tasks set using learned predictive FEF scheduling is 1954.13 milliseconds.

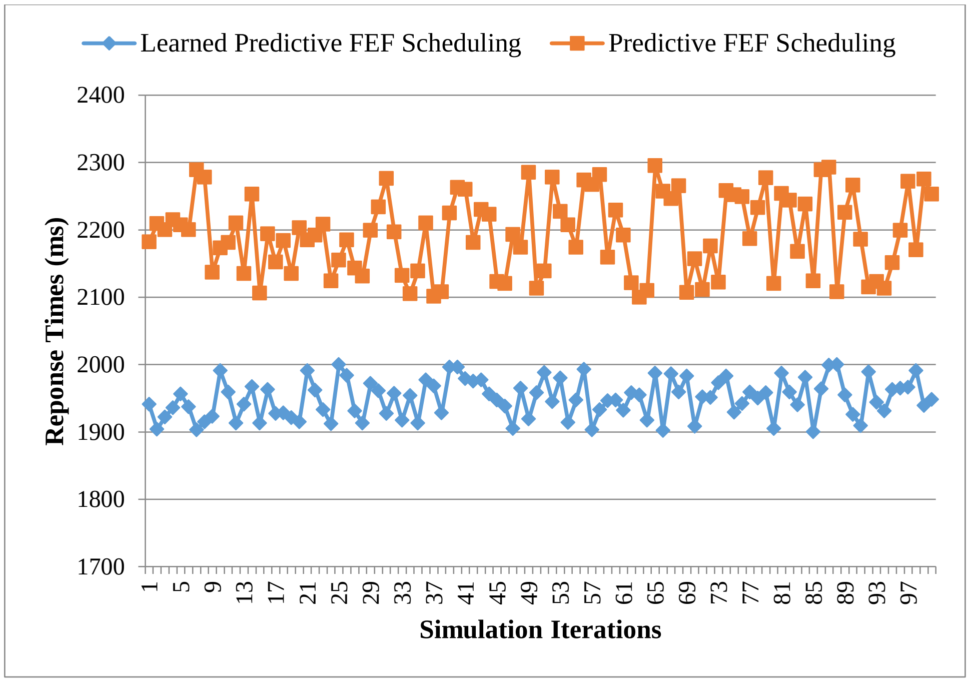


Figure 41: Response time comparisons with and without learned prediction for candy box factory

In the Figure 42, we present the comparisons among learned predictive FEF scheduling and learned predictive hybrid scheduling. The graph shows the tasks average response times at y-axis and test iterations at x-axis. The hybrid scheduling mechanism is the combination of ACM and FEF scheduling approach. The hybrid mechanism improves the overall task allocation to machine pairs and hence also improves the response time as seen in the results. The average response time for tasks set using learned predictive FEF scheduling is 1954.13 milliseconds and the average response time for tasks set using learned predictive hybrid scheduling is 1769.69 milliseconds.

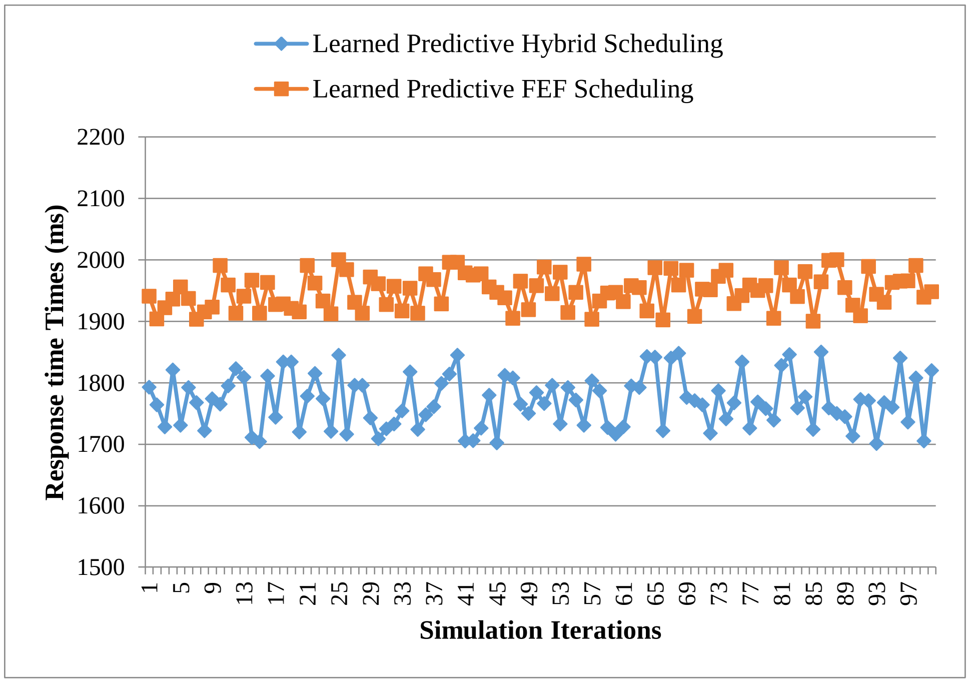


Figure 42: Response time comparisons with learned predictive FEF and learned predictive hybrid scheduling for candy box factory

In the Figure 43, we present the comparisons among learned predictive hybrid scheduling and learned predictive and optimized hybrid scheduling. The graph shows the tasks average execution times at y-axis and test iterations at x-axis. The goal of optimization module is to maximize the utilization of machine network and machine pairs to which tasks are allocated.

Hence the learned predictive and optimized hybrid scheduling results in better performance by improving the machine utilization and improving the response time. The average response time for tasks set using learned predictive hybrid scheduling is 1769.69 milliseconds and the average response time for tasks set using learned predictive and optimized hybrid scheduling is 1555.4 milliseconds.

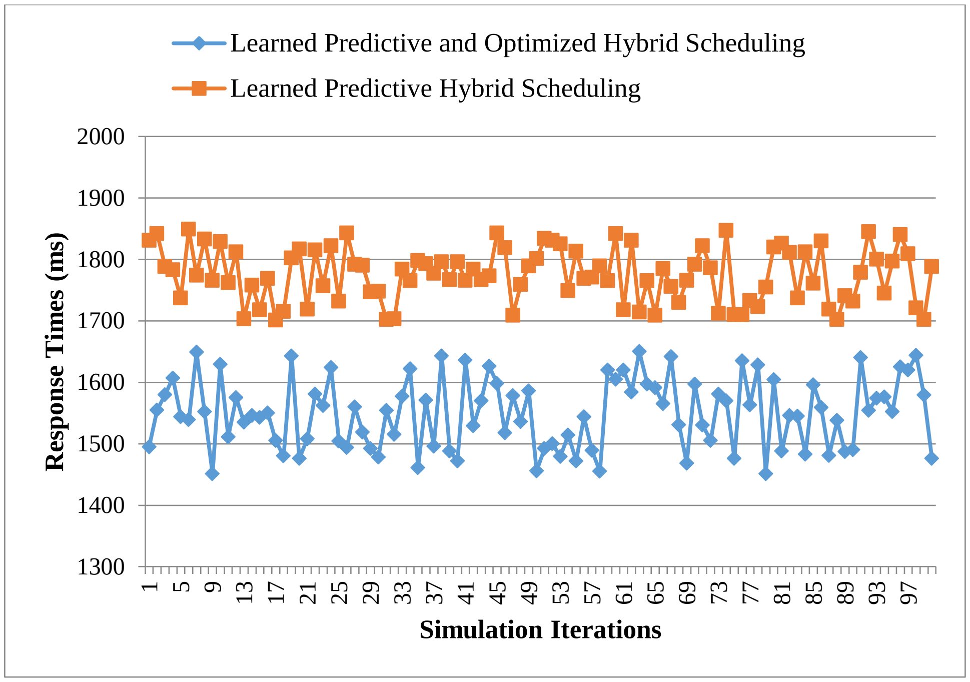


Figure 43: Response time comparisons with and without optimized scheduling for candy box factory

The Figure 44 shows the assembly machines network utilization rate for tasks scheduling simulations. There are four assembly machines as AM1, AM2, AM3, and AM4. The graph below shows average machine utilization for each machine in the assembly machine network. The comparisons are drawn between proposed hybrid scheduling approach based on learning to prediction and optimization with predictive FEF scheduling scheme. The learned scheduling approach increases the machine utilization as it takes aid from the learning modules of prediction and optimization to make intelligent scheduling decisions and increases the overall system performance.

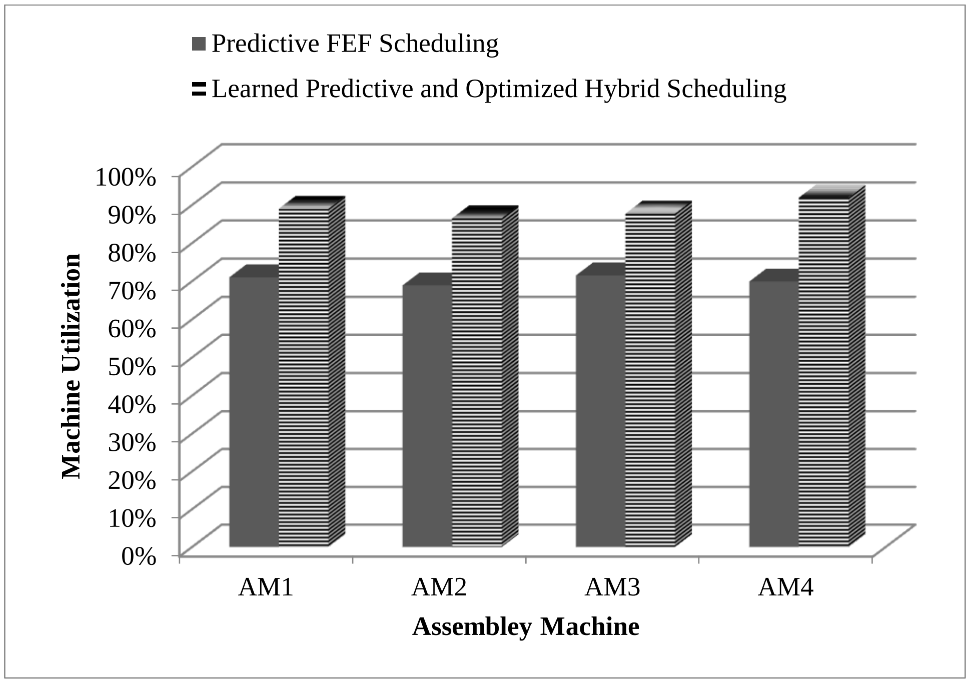
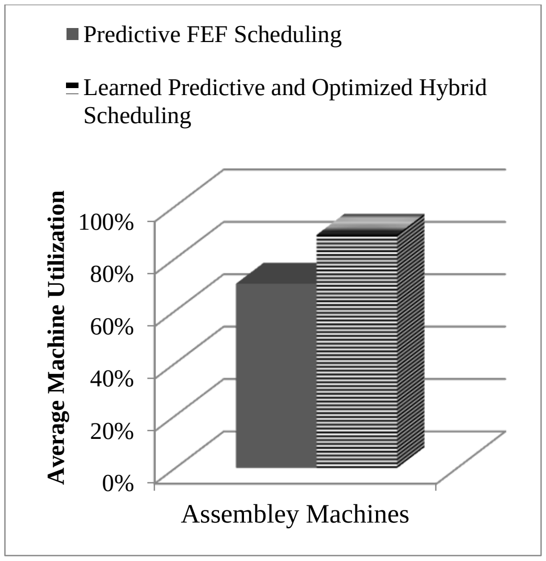
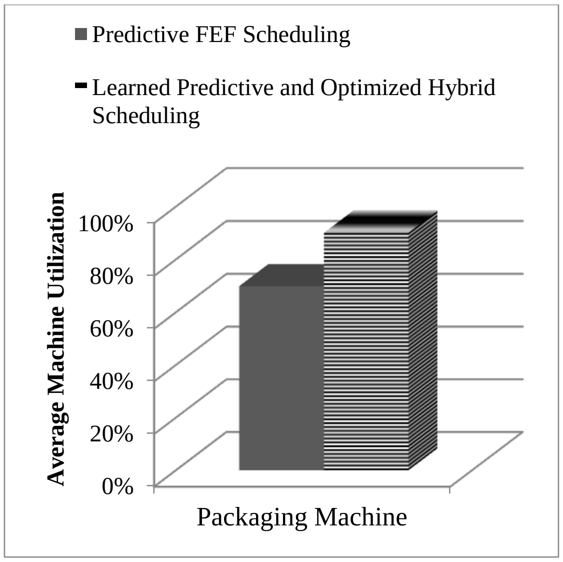


Figure 44: Machine utilization rate comparisons for proposed scheduling schemes in assembly machine in candy box factory





(a) **(b)**

Figure 45: Average machine utilization for proposed scheduling schemes in candy box factory (a) for assembly machine; (b) for packaging machine

Similarly, the machine utilization for the machines in the packaging machines networks is observed to be higher in learned predictive and optimized hybrid scheduling in comparison to the predictive FFF scheduling. The Figure 45 shows the overall average machine utilization rate for each assembly network and packaging network; based on the comparisons among the predictive FEF scheduling with learned predictive and optimized hybrid scheduling. The average machine utilization for predictive FEF scheduling is 70% and the average machine utilization for predictive and optimized hybrid scheduling is 89% for assembly machines. The average machine utilization for predictive FEF scheduling is 69.90% and the average machine utilization for predictive and optimized hybrid scheduling is 90% for packaging machines. The results prove that learned predictive and optimized hybrid scheduling shows far better performance in comparison to the predictive FEF scheduling.

5.3 Simulation and Performance Analysis of Simulated Tasks Dataset

5.3.1 Input Tasks Modeling for Data Simulations

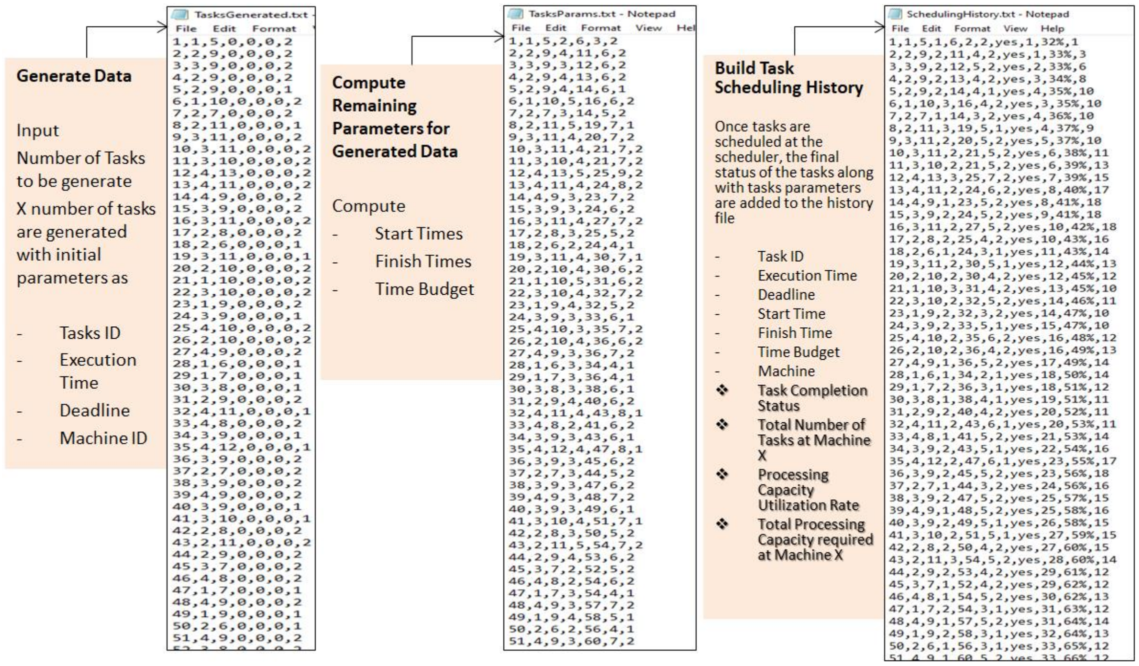
In this sub-section, we present the input tasks modeling for simulated tasks dataset. The input tasks for simulated tasks dataset are randomly generated based on user inputs and system thresholds to generate tasks set to be simulated. The tasks generated have initial parameters as tasks ID, execution time, and deadline and machine ID. Next the system computes the tasks parameters as start time, finish time, and time budget time based on initially generated parameters. Table 20 below shows the list of tasks parameters generated for the tasks.

Table 20: Generated Task Parameters List

Task Parameter	**Description**
Task ID	Tasks’ unique identifier
Arrival Time	Task arrival time at system
Execution Time	Time required to complete task
Deadline	Time before which task should complete
Machine ID	Machine at which task is to be executed
Start Times	Tasks’ start time at machine
Finish Time	Tasks’ finish time at machine

At first, the number of tasks to be generated is taken as input from the user. Next the tasks generation interval is taken as input from the user. The generated tasks are sensing tasks with different sensing intervals as 5 seconds, 10 seconds, 15 seconds, 20 seconds, 30 seconds, 40 seconds and 60 seconds. For each task, task id are generated, arrival time is time at which task arrived to at system, next the execution times are randomly generated between a given range, task deadline are generated between given threshold of being greater than zero and less than the tasks deadline, machine id is initialized as zero and later set to the scheduled machine, start times are set to the scheduled start times at machine, finish time is set to the scheduled finish time at machine.

In the first step for tasks generation, the sensing tasks are generated based on the initial parameters for task id, execution time, and deadline and machine id. In the next step the tasks parameter values generation function for start times assignment, finish time assignment and time budget assignment are called where these parameters are initialized. Using these parameters, the tasks are ready to be run at the scheduler at their scheduled time following scheduling mechanism. In parallel to tasks scheduling at the scheduler, the scheduler keeps maintaining the history logs with tasks detailed parameters and additional parameters of task completion status, total number tasks at each machine with machine id, processing capacity of each machine with machine id, total processing capacity required by each machine based on current load (Figure 46).


Generate Data
Input
Number of Tasks to be generate
X number of tasks are generated with initial parameters as
- Tasks ID
- Execution Time
- Deadline
- Machine ID
TasksGenerated.txt
File Edit Format
Compute Remaining Parameters for Generated Data
Compute
- Start Times
- Finish Times
- Time Budget
TasksParams.txt - Notepad
File Edit Format View Hel
Build Task Scheduling History
Once tasks are scheduled at the scheduler, the final status of the tasks along with tasks parameters are added to the history file
- Task ID
- Execution Time
- Deadline
- Start Time
- Finish Time
- Time Budget
- Machine
❖ Task Completion Status
❖ Total Number of Tasks at Machine X
❖ Processing Capacity Utilization Rate
❖ Total Processing Capacity required at Machine X
SchedulingHistory.txt - Notepad
File Edit Format View Help

```
1,1,5,0,0,0,2
2,2,9,0,0,0,2
3,3,9,0,0,0,2
4,2,9,0,0,0,2
5,2,9,0,0,0,1
6,1,10,0,0,0,2
7,2,7,0,0,0,2
8,2,11,0,0,0,1
9,3,11,0,0,0,2
10,3,11,0,0,0,2
11,3,10,0,0,0,2
12,4,13,0,0,0,2
13,4,11,0,0,0,2
14,4,9,0,0,0,2
15,3,9,0,0,0,2
16,3,11,0,0,0,2
17,2,8,0,0,0,2
18,2,6,0,0,0,1
19,3,11,0,0,0,1
20,2,10,0,0,0,2
21,1,10,0,0,0,2
22,3,10,0,0,0,2
23,1,9,0,0,0,2
24,3,9,0,0,0,1
25,4,10,0,0,0,2
26,2,10,0,0,0,2
27,4,9,0,0,0,2
28,1,6,0,0,0,1
29,1,7,0,0,0,1
30,3,8,0,0,0,1
31,2,9,0,0,0,2
32,4,11,0,0,0,1
33,4,8,0,0,0,2
34,3,9,0,0,0,1
35,4,12,0,0,0,1
36,3,9,0,0,0,2
37,2,7,0,0,0,2
38,3,9,0,0,0,2
39,4,9,0,0,0,2
40,3,9,0,0,0,1
41,3,10,0,0,0,1
42,2,8,0,0,0,2
43,2,11,0,0,0,2
44,2,9,0,0,0,2
45,3,7,0,0,0,2
46,4,8,0,0,0,2
47,1,7,0,0,0,1
48,4,9,0,0,0,2
49,1,9,0,0,0,1
50,2,6,0,0,0,1
51,4,9,0,0,0,2
```

```
1,1,5,2,6,3,2
2,2,9,4,11,6,2
3,3,9,3,12,6,2
4,2,9,4,13,6,2
5,2,9,4,14,6,1
6,1,10,5,16,6,2
7,2,7,3,14,5,2
8,2,11,5,19,7,1
9,3,11,4,20,7,2
10,3,11,4,21,7,2
11,3,10,4,21,7,2
12,4,13,5,25,9,2
13,4,11,4,24,8,2
14,4,9,3,23,7,2
15,3,9,3,24,6,2
16,3,11,4,27,7,2
17,2,8,3,25,5,2
18,2,6,2,24,4,1
19,3,11,4,30,7,1
20,2,10,4,30,6,2
21,1,10,5,31,6,2
22,3,10,4,32,7,2
23,1,9,4,32,5,2
24,3,9,3,33,6,1
25,4,10,3,35,7,2
26,2,10,4,36,6,2
27,4,9,3,36,7,2
28,1,6,3,34,4,1
29,1,7,3,36,4,1
30,3,8,3,38,6,1
31,2,9,4,40,6,2
32,4,11,4,43,8,1
33,4,8,2,41,6,2
34,3,9,3,43,6,1
35,4,12,4,47,8,1
36,3,9,3,45,6,2
37,2,7,3,44,5,2
38,3,9,3,47,6,2
39,4,9,3,48,7,2
40,3,9,3,49,6,1
41,3,10,4,51,7,1
42,2,8,3,50,5,2
43,2,11,5,54,7,2
44,2,9,4,53,6,2
45,3,7,2,52,5,2
46,4,8,2,54,6,2
47,1,7,3,54,4,1
48,4,9,3,57,7,2
49,1,9,4,58,5,1
50,2,6,2,56,4,1
51,4,9,3,60,7,2
```

```
1,1,5,1,6,2,2,yes,1,32%,1
2,2,9,2,11,4,2,yes,1,33%,3
3,3,9,2,12,5,2,yes,2,33%,6
4,2,9,2,13,4,2,yes,3,34%,8
5,2,9,2,14,4,1,yes,4,35%,10
6,1,10,3,16,4,2,yes,3,35%,10
7,2,7,1,14,3,2,yes,4,36%,10
8,2,11,3,19,5,1,yes,4,37%,9
9,3,11,2,20,5,2,yes,5,37%,10
10,3,11,2,21,5,2,yes,6,38%,11
11,3,10,2,21,5,2,yes,6,39%,13
12,4,13,3,25,7,2,yes,7,39%,15
13,4,11,2,24,6,2,yes,8,40%,17
14,4,9,1,23,5,2,yes,8,41%,18
15,3,9,2,24,5,2,yes,9,41%,18
16,3,11,2,27,5,2,yes,10,42%,18
17,2,8,2,25,4,2,yes,10,43%,16
18,2,6,1,24,3,1,yes,11,43%,14
19,3,11,2,30,5,1,yes,12,44%,13
20,2,10,2,30,4,2,yes,12,45%,12
21,1,10,3,31,4,2,yes,13,45%,10
22,3,10,2,32,5,2,yes,14,46%,11
23,1,9,2,32,3,2,yes,14,47%,10
24,3,9,2,33,5,1,yes,15,47%,10
25,4,10,2,35,6,2,yes,16,48%,12
26,2,10,2,36,4,2,yes,16,49%,13
27,4,9,1,36,5,2,yes,17,49%,14
28,1,6,1,34,2,1,yes,18,50%,14
29,1,7,2,36,3,1,yes,18,51%,12
30,3,8,1,38,4,1,yes,19,51%,11
31,2,9,2,40,4,2,yes,20,52%,11
32,4,11,2,43,6,1,yes,20,53%,11
33,4,8,1,41,5,2,yes,21,53%,14
34,3,9,2,43,5,1,yes,22,54%,16
35,4,12,2,47,6,1,yes,23,55%,17
36,3,9,2,45,5,2,yes,23,56%,18
37,2,7,1,44,3,2,yes,24,56%,16
38,3,9,2,47,5,2,yes,25,57%,15
39,4,9,1,48,5,2,yes,25,58%,16
40,3,9,2,49,5,1,yes,26,58%,15
41,3,10,2,51,5,1,yes,27,59%,15
42,2,8,2,50,4,2,yes,27,60%,15
43,2,11,3,54,5,2,yes,28,60%,14
44,2,9,2,53,4,2,yes,29,61%,12
45,3,7,1,52,4,2,yes,29,62%,12
46,4,8,1,54,5,2,yes,30,62%,13
47,1,7,2,54,3,1,yes,31,63%,12
48,4,9,1,57,5,2,yes,31,64%,14
49,1,9,2,58,3,1,yes,32,64%,13
50,2,6,1,56,3,1,yes,33,65%,12
51,4,9,1,60,5,2,yes,33,66%,12
```

Figure 46: Tasks parameters generation and building history data parameters

5.3.2 Performance Analysis and Comparisons

In this part, we have used the ANN based prediction algorithm. The prediction algorithm is then learned by optimizing the ANN weights using PSO algorithm. The addition of PSO in ANN improves the performance of prediction algorithm greatly and gives better prediction accuracies with fewer numbers of iterations.

In Figure 47, we present the performance results comparison for prediction accuracies based on the ANN algorithm and PSO learning based ANN algorithm (PSO-NN). The maximum prediction accuracy achieved using ANN is 99.32% in 600 iterations while the PSO-NN achieves the prediction accuracy of 99.36% in 600 iterations and the maximum prediction accuracy achieved using PSO-NN is 99.42% in 800 iterations. The results clearly show that PSO-NN outperforms the ANN prediction accuracy results. It achieves higher accuracy with less number of iterations.

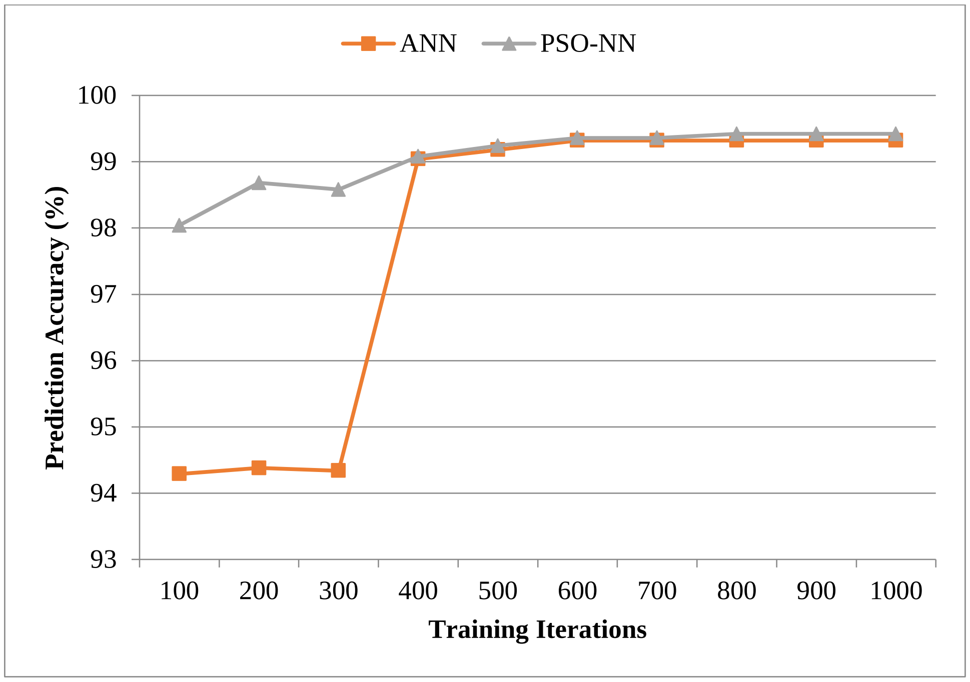


Figure 47: Prediction accuracy comparisons for ANN and PSO-NN in simulated tasks dataset

Next, we compare the prediction accuracy results with prediction learning based on the proposed variations of the PSO algorithm as R-PSO and VB-PSO. The Figure 48 shows the output results for learned prediction with optimized weights of ANN based on PSO and its variations ad R-PSO and VB-PSO. The results show that R-PSO and VB-PSO, both achieve higher accuracy within less iterations in comparison to the PSO so both variations are considered fruitful improvements in PSO. In comparing R-PSO and VB-PSO, we can observe that initially R-PSO-NN achieved higher prediction accuracy than VB-PSO but within next 100 iterations the VB-PSO-NN's prediction accuracy shoots higher and stays to 99.54% from 200 iterations onwards. Whereas, though R-PSO-NN takes 600 iterations to achieve prediction accuracy of 99.51% and reaches to 99.69% of prediction accuracy 700 iterations and gives higher prediction accuracy eventually.

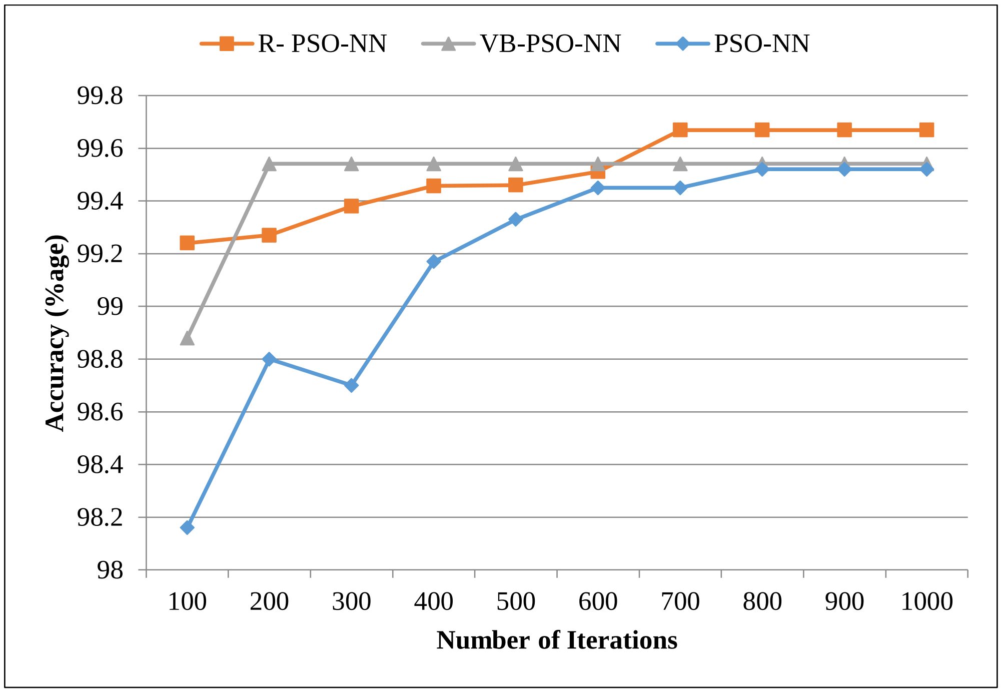
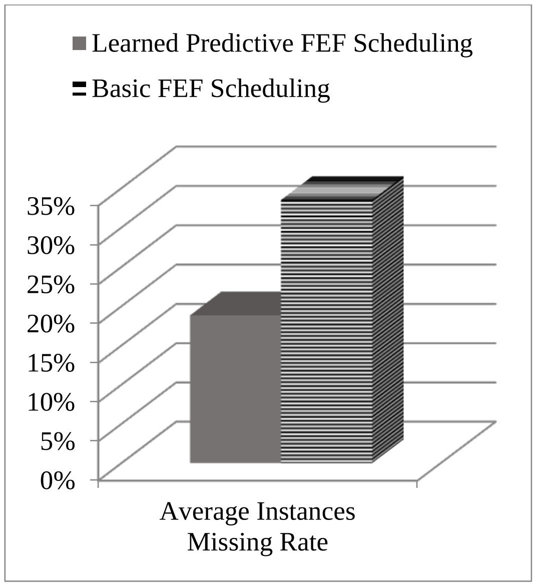
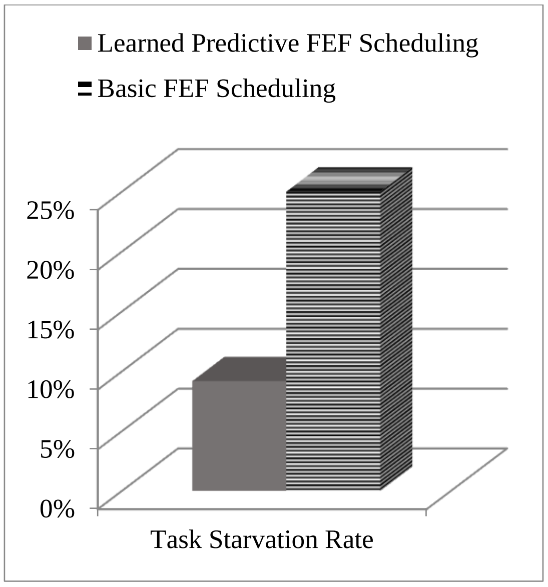


Figure 48: Prediction accuracy comparisons based on PSO-NN, R-PSO-NN and VB-PSO-NN in simulated tasks dataset

In Figure 49, we present the comparison of baseline FEF with learned prediction FEF with aim to demonstrate the effect of learning to prediction in the scheduling algorithm. The Figure shows the results for tasks starvation rate and average instances missing rate. We can observe that the learned predictive FEF reduces the tasks starvation rate and also reduces the average instances missing rate. The average instances missing rate for predictive FEF scheduling 33.49% and that for learned predictive and optimized hybrid scheduling is 19%. The tasks starvation rate for predictive FEF scheduling 25% and that for learned predictive and optimized hybrid scheduling is 9.19%. The learned predictive FEF decreases the tasks starvation rate with wisely allocating the free machine slots. Hence, the learning to prediction module increases the overall performance of the scheduler.





(a) **(b)**

Figure 49: Comparisons for learned predictive FEF and basic FEF scheduling for simulated tasks dataset (a) average instances missing rate in percentage; (b) average tasks starvation rate in percentage

Now, we evaluate the effect of learning to optimization module. The optimization module works with an objective of maximizing the machine utilization. It takes the current tasks instances as input and finds the best orders and pairs for tasks to be processed at machines. The objective function will struggle to find orders with maximum machine utilization and pass the orders back to the scheduling module. We compare the learned predictive hybrid scheduling with learned prediction and optimized hybrid scheduling in Figure 50. The average machine utilization rate for learned predictive hybrid scheduling is 72.59% and the average machine utilization rate for learned predictive and optimized scheduling is 90.59%. In the results, we can observe that optimization based task scheduling increases the machine utilization in comparison the one without optimization.

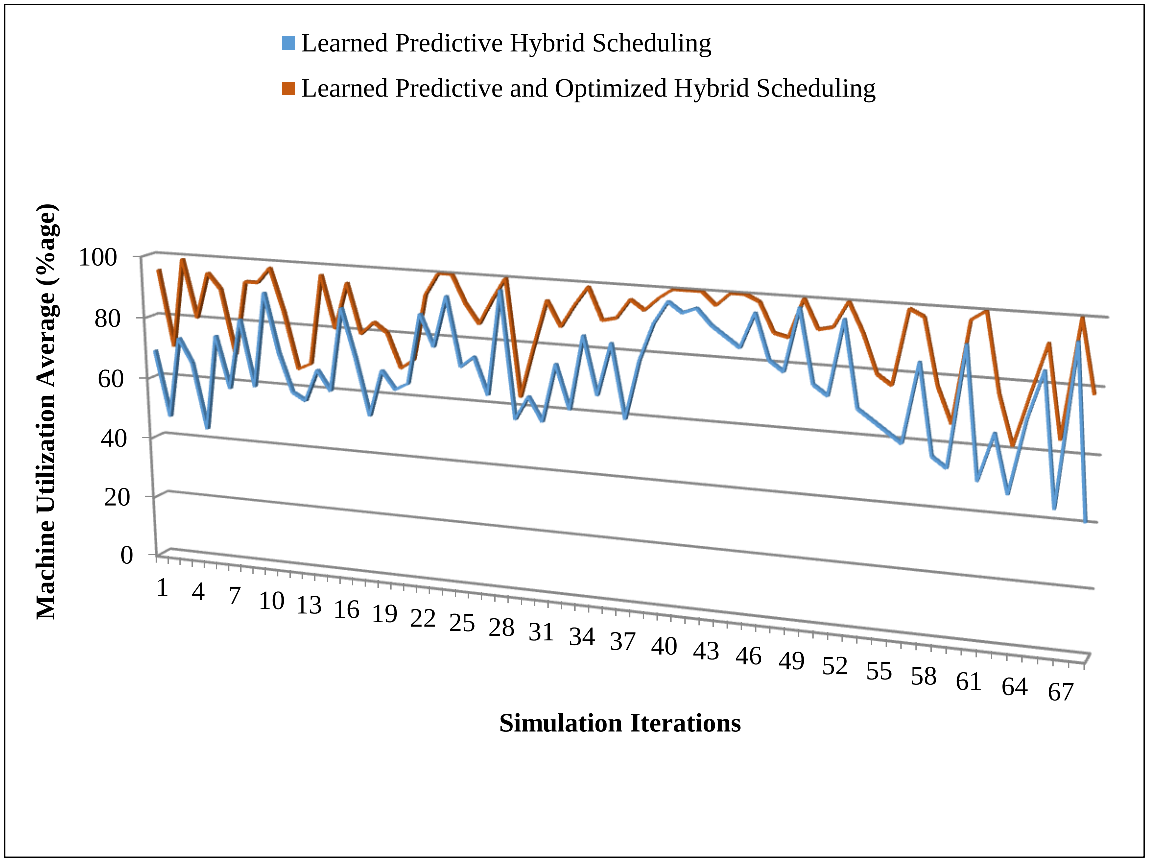


Figure 50: Machine utilization comparisons with optimization vs. without optimization for simulated tasks dataset

The Figure 51 shows the comparisons analysis for average response time taken for predictive FEF scheduling, learned FEF scheduling, learned predictive hybrid scheduling, and learned predictive and optimized hybrid scheduling scheme. We can observe from the comparisons that as the learning module of prediction is added, a significant decrease in the response times is noticed. Next, with the change in scheduling algorithm from FEF to hybrid approach, the performance gets slightly better and further decrease is observed in the response time. Finally, with the addition of learning to optimization module, as the machine utilization increases, hence the task waiting time decreases and response time taken also decreases.

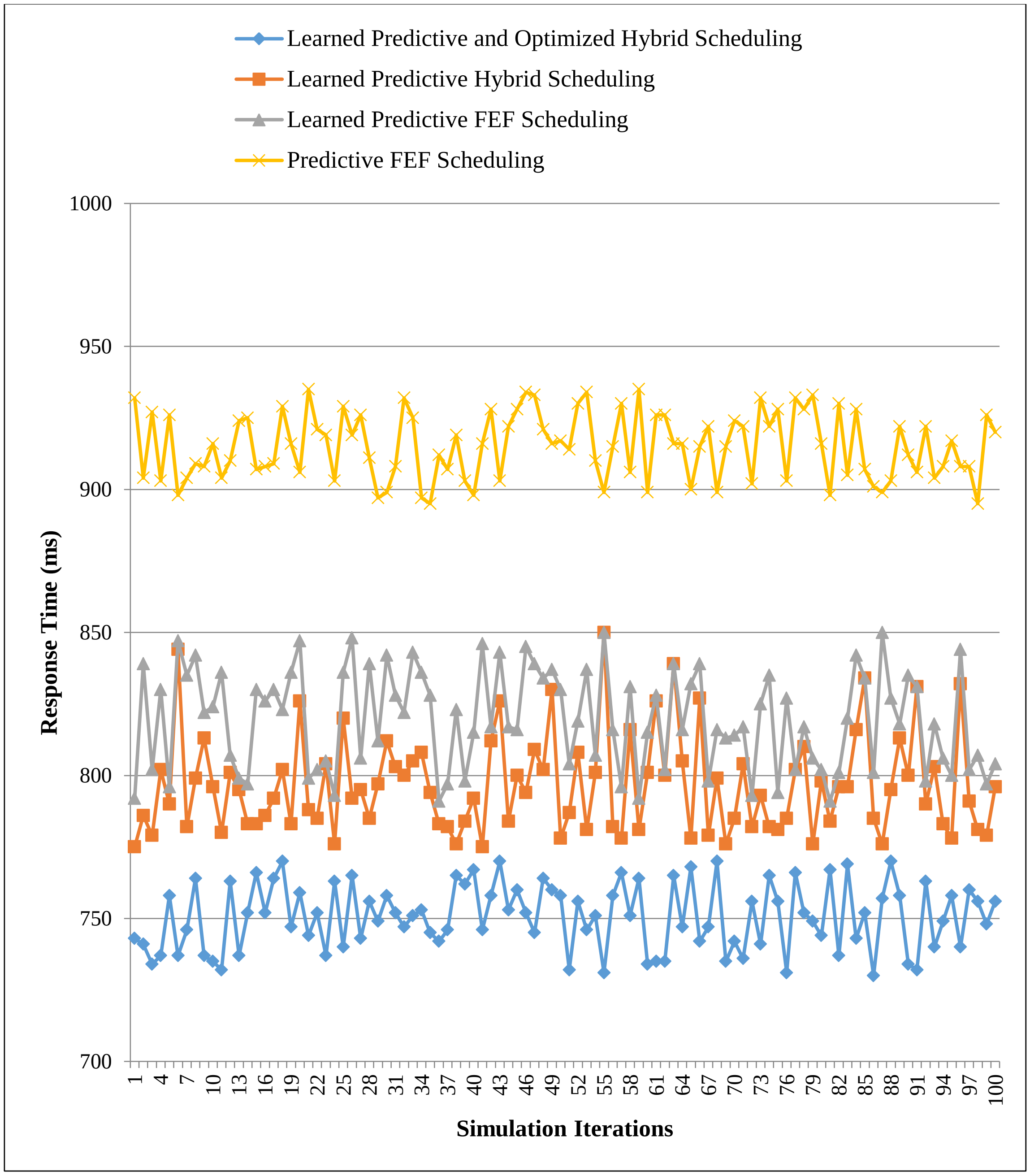


Figure 51: Response time comparisons of proposed scheduling schemes for simulated tasks dataset

The average response time for tasks set using predictive FEF scheduling is 915.64 milliseconds, using learned predictive FEF scheduling is 820.34 milliseconds, using learned predictive hybrid scheduling is 797.59 milliseconds and using learned predictive and optimized hybrid scheduling is 750.66 milliseconds.

The Figure 52 shows the average instances missed rate for the predictive FEF scheduling, learned predictive FEF scheduling, learned predictive hybrid scheduling and learned predictive and optimized hybrid scheduling. The average instances missing rate for tasks set using predictive FEF scheduling is 36%, using learned predictive FEF scheduling is 19%, using learned predictive hybrid scheduling is 10% and using learned predictive and optimized hybrid scheduling is 8%. In the results shown, we can observe that the least number of tasks missing rate is at learned predictive and optimized hybrid scheduling, as it includes the learning modules and has the maximum machine utilization rate.

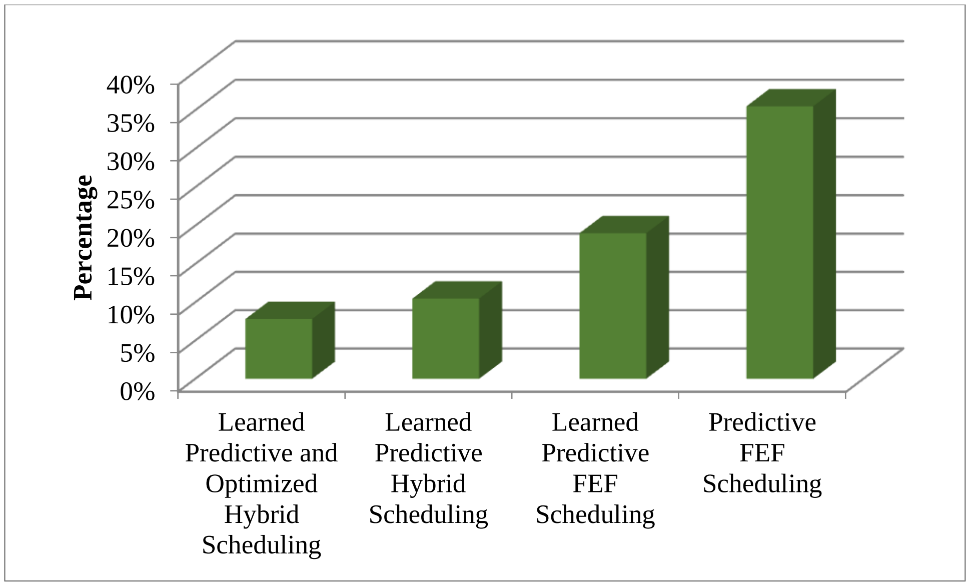


Figure 52: Average instances missing rate comparisons of proposed scheduling schemes for simulated tasks dataset

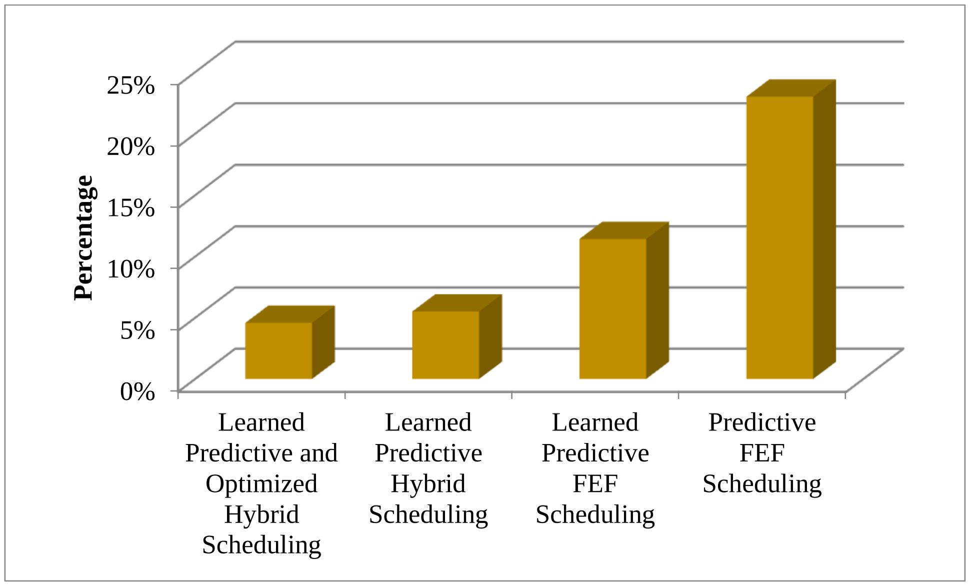


Figure 53: Average tasks starvation rate comparisons of proposed scheduling schemes for simulated tasks dataset

The Figure 53 shows the average tasks starvation rate for the predictive FEF scheduling, learned predictive FEF scheduling, learned predictive hybrid scheduling and learned predictive and optimized hybrid scheduling. The average instances starvation rate for tasks set using predictive FEF scheduling is 23%, using learned predictive FEF scheduling is 11%, using learned predictive hybrid scheduling is 6% and using learned predictive and optimized hybrid scheduling is 5%. In the results shown, we can observe that the least number of tasks starvation rate is at learned predictive and optimized hybrid scheduling, as it includes the learning modules and has the maximum machine utilization rate.

5.4 Simulation and Performance Analysis of Machine Cluster Data

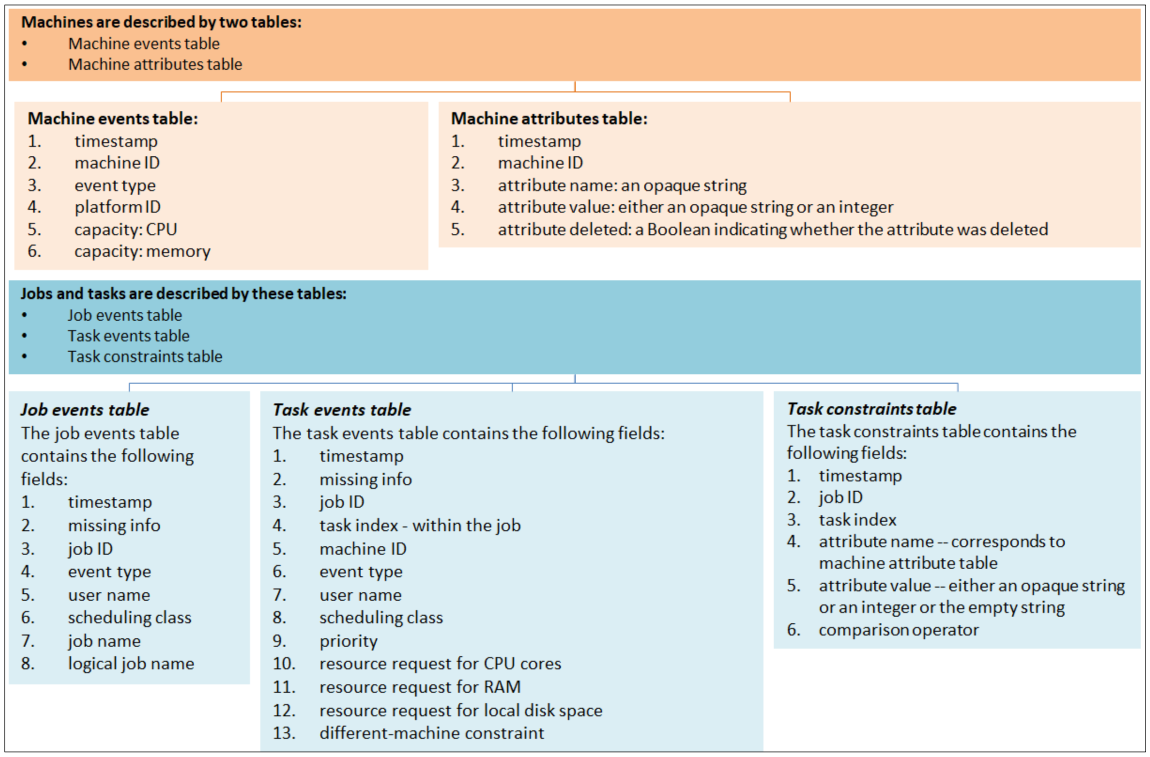
5.4.1 Input Dataset

In this section, we have used google cloud task scheduling dataset [55] for the simulations and performance evaluations of our system. The dataset compromises of 500 sets of tasks instances executed at multiple machines. Table 21 shows the data size.

Table 21: Machine cluster dataset size

Dataset Table	Size
Machine Attributes	1048576
Machine Events	37780
Total Task Sets	500
Single Task Set Instances	1048576

The dataset has two main data as machine data and tasks data. Each task comprises of multiple jobs and includes jobs data. The machine data has two main tables as machine events table and machine attributes table. The machine events table contains timestamp, machine ID, event type, platform ID, machine processing capacity and machine memory capacity. Machine attributes table contains timestamp, machine ID, and attribute name, value and deletion status. The tasks data has tasks events table, tasks constraints table and jobs events table. The jobs events table contains time stamp, missing information, job ID, event type, user name, scheduling class and job name. The tasks event table contains timestamp, missing information, job ID, task index for job, machine ID, event type, scheduling class, priority, resources for CPU, RAM and memory, and machine constraints. The tasks constraints table contains timestamp job ID, task index, attribute name and value.



Machines are described by two tables:
- Machine events table
- Machine attributes table

Machine events table:
1. timestamp
2. machine ID
3. event type
4. platform ID
5. capacity: CPU
6. capacity: memory

Machine attributes table:
1. timestamp
2. machine ID
3. attribute name: an opaque string
4. attribute value: either an opaque string or an integer
5. attribute deleted: a Boolean indicating whether the attribute was deleted

Jobs and tasks are described by these tables:
- Job events table
- Task events table
- Task constraints table

Job events table
The job events table contains the following fields:
1. timestamp
2. missing info
3. job ID
4. event type
5. user name
6. scheduling class
7. job name
8. logical job name

Task events table
The task events table contains the following fields:
1. timestamp
2. missing info
3. job ID
4. task index - within the job
5. machine ID
6. event type
7. user name
8. scheduling class
9. priority
10. resource request for CPU cores
11. resource request for RAM
12. resource request for local disk space
13. different-machine constraint

Task constraints table
The task constraints table contains the following fields:
1. timestamp
2. job ID
3. task index
4. attribute name -- corresponds to machine attribute table
5. attribute value -- either an opaque string or an integer or the empty string
6. comparison operator

Figure 54: Detailed hierarchy view for in machine cluster dataset

5.4.2 Performance Analysis and Comparisons

In this sub-section, we present the experiments and performance analysis for the machine cluster data as input.

First of all, the machine cluster data is used to train the prediction model. The predictions are made using ANNs where ANNs' weights are tuned using PSO variations. In the Figure 55 below, we show the prediction accuracy achieved and comparisons of the accuracy among implementations of PSO based ANN predictions, R-PSO based ANN predictions and VB-PSO based ANN predictions. In the graph, we observe that VB-PSO achieves the highest prediction accuracy within least number of epochs, whereas though R-PSO-NN also achieves the same accuracy as VB-PSO-NN but with higher number of epochs. The prediction accuracy of 98.42% is achieved by R-PSO-NN in 800 iterations while VB-PSO-NN achieves the same within 300 iterations.

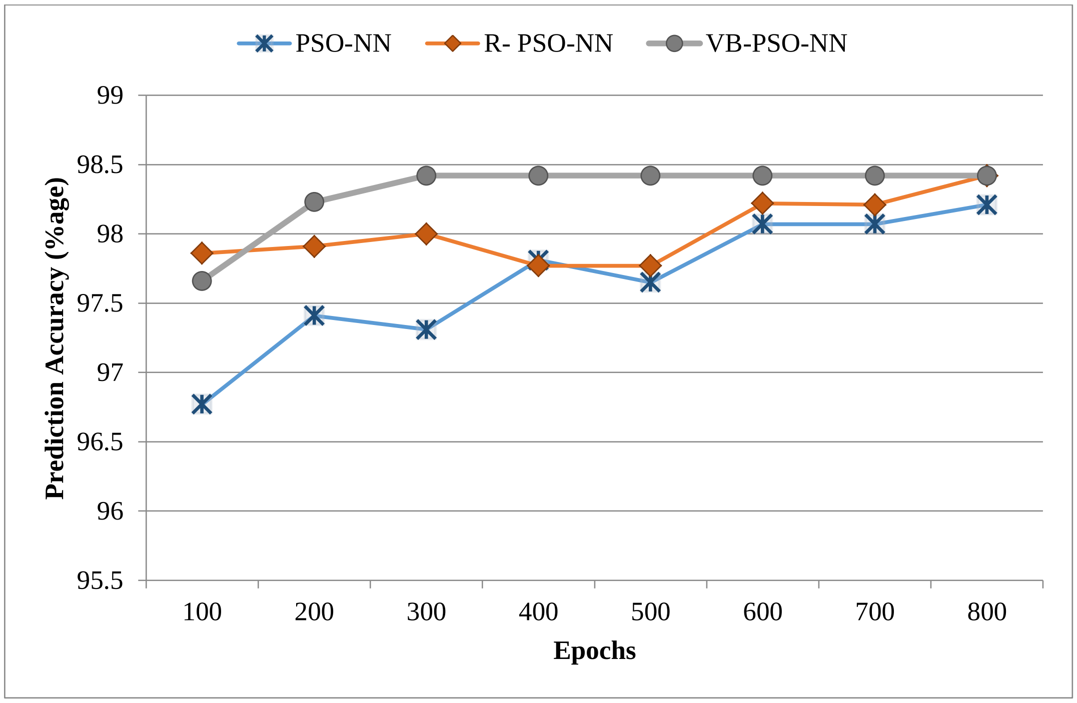
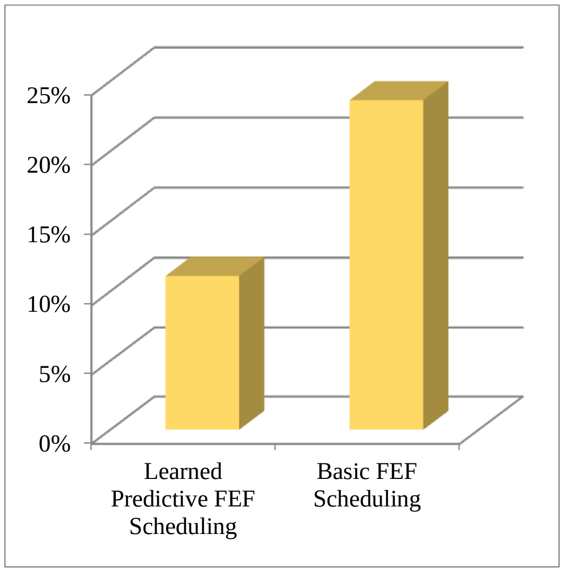
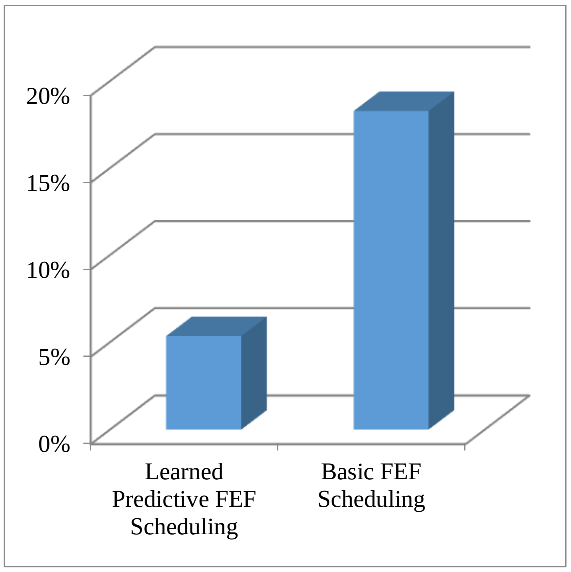


Figure 55: Prediction accuracy comparisons based on PSO-NN, R-PSO-NN and VB-PSO-NN in machine cluster dataset

In Figure 56, we present the comparison of basic FEF scheduling and learned prediction FEF scheduling. The learned predictive FEF scheduling has an addition prediction module based on ANN which is learned using PSO. The learned prediction enhances the scheduling performance using history data learning and optimization of prediction results using PSO to tune ANN's weights. Hence, we can observe in the graph that learned predictive FEF has less number of tasks starved and less number of instances missed in comparison to basic FEF scheduling. The average instances missing rate for basic FEF scheduling is 24% and for learned predictive FEF scheduling is 11%. The average tasks starvation rate for basic FEF scheduling is 18.28% and for learned predictive FEF scheduling is 5.35%.





(a) **(b)**

Figure 56: Comparisons for learned predictive FEF and basic FEF scheduling in machine cluster dataset (a) average instances missing rate in percentage; (b) average tasks starvation rate in percentage

In the Figure 57, we compare the predictive FEF scheduling with learned predictive and optimized hybrid scheduling. The graph shows the machine utilization rate based on varying tasks load. The tasks generated vary from 100 to 1000 during the simulation period. As the tasks grow more than the total machine capacity, some of the tasks instances are must to drop out. In basic FEF scheduling, the tasks instance missing rate is observed higher even when the machine is not being fully utilized, while in the learned predictive and optimized hybrid scheduling the tasks instances are only missed when the machine are in use to the fullest of their capacity. Also, in learned predictive and optimized hybrid scheduling the tasks being processed are high priority tasks and scheduler will make sure to allocate resources based on priorities, in order to not miss any high priority task when it can be traded off with a less priority task.

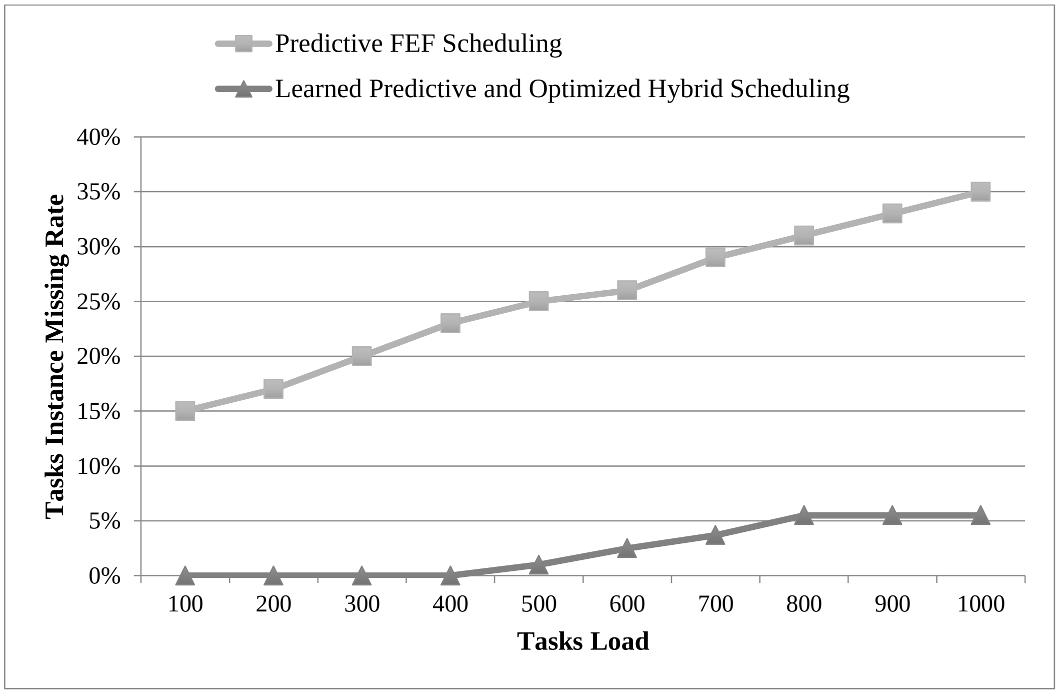


Figure 57: Tasks instances missing rate comparisons for proposed schemes with varying number of tasks in machine cluster dataset

In the Figure 58, the average machine utilization for predictive FEF scheduling is compared with learned predictive and optimized hybrid scheduling. The machine utilization is measured with respect to varying number of available machines. In the results, we can observe that machine utilization increases with the increase in the tasks; however, the learned scheduling has high machine utilization in comparison to the predictive FEF scheduling. This indicates that learned predictive and optimized hybrid scheduling executes a higher number of tasks successfully while predictive FEF scheduling might be missing some tasks instances with high starvation rate with machine slots not being utilized to their full potential.

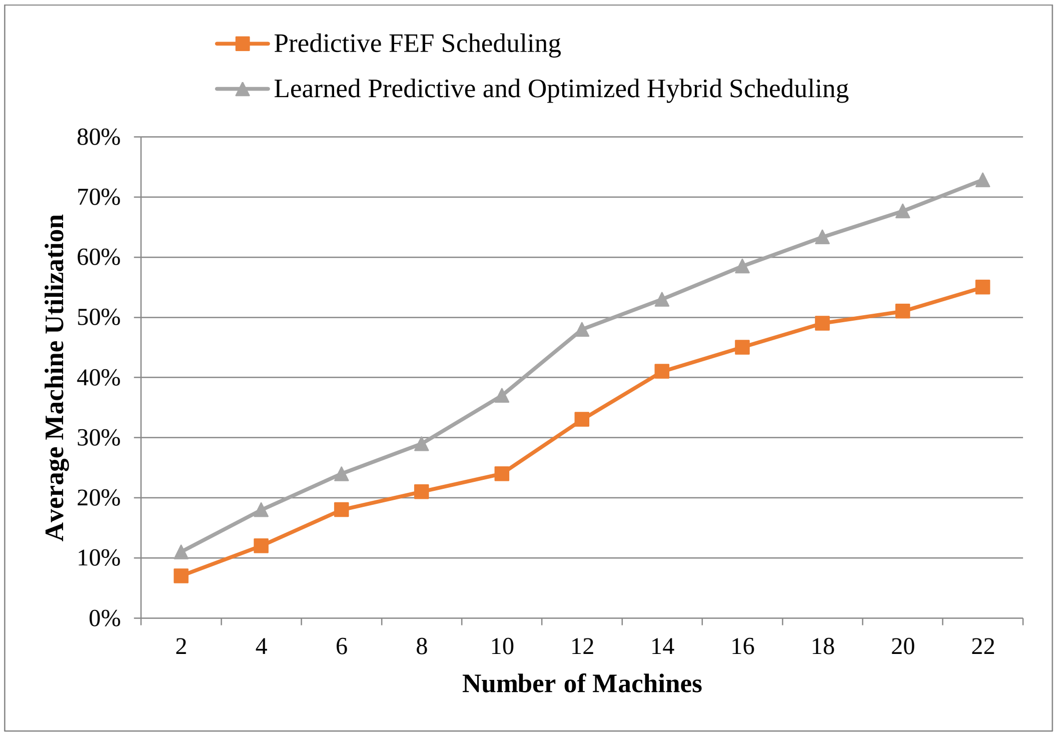


Figure 58: Average machine utilization rate comparisons for proposed schemes with varying number of tasks in machine cluster dataset

In the Figure 59, we present the comparisons among predictive FEF scheduling, learned predictive FEF scheduling, learned predictive hybrid scheduling and, learned predictive and optimized hybrid scheduling. The x-axis shows the average response time of machine for tasks and y-axis shows the test iterations. First, we analyze the predictive FEF scheduling and learned predictive FEF scheduling. The addition of learning in the prediction module substantially improves the performance and machine response time is reduced. Next, we evaluate the learned predictive FEF scheduling and learned predictive hybrid scheduling. It majorly reflects the difference in mechanism based on replacing the FEF scheduling algorithm to the hybrid ACM-FEF scheduling algorithm.

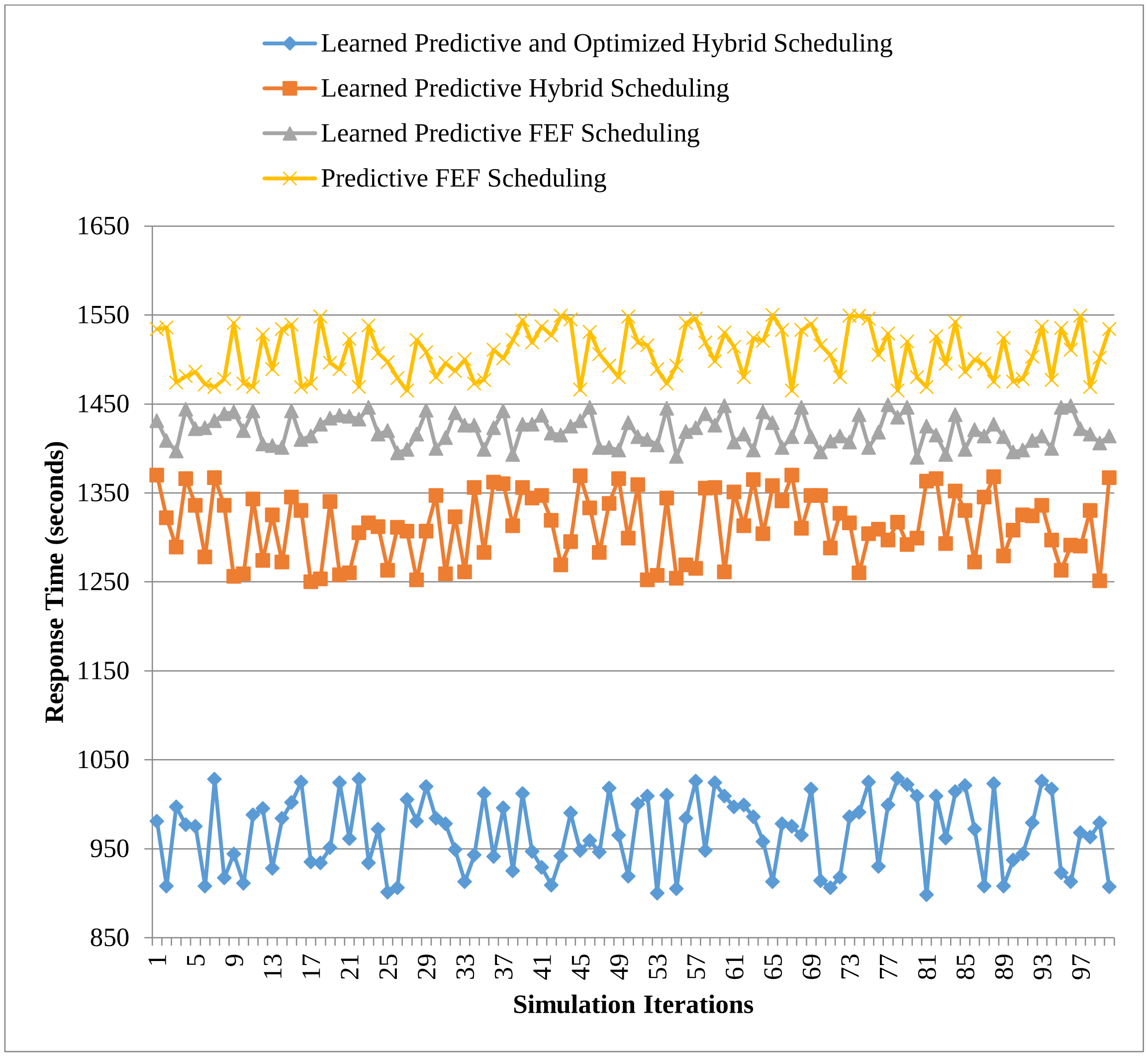


Figure 59: Average machine utilization rate comparisons for proposed schemes with varying number of machines in machine cluster dataset

We can observe that hybrid approach brings an improvement in the performance by further reducing the response time. At last we evaluate the learned predictive hybrid scheduling and learned predictive and optimized hybrid scheduling. It highlights the performance of optimization module which is based on the objective of maximizing the machine utilization as a single object as well as maximizing the machine network utilization overall. The addition of optimization module also improves the performance to some extent as observed in the results. The average response time for tasks set using predictive FEF scheduling is 1505.45

milliseconds, using learned predictive FEF scheduling is 1416.99 milliseconds, using learned predictive hybrid scheduling is 1310.85 milliseconds and using learned predictive and optimized hybrid scheduling is 957.8 milliseconds.

Chapter 6: Conclusions

Efficient real-time tasks scheduling based on predictive analytics and optimized techniques is vital for smart factory systems. In this thesis, we have proposed an integrated solution for smart factory's efficient task management based on learning to scheduling. The proposed proposal provides integrated task management solution based on learning to prediction and learning to optimization techniques. The proposed system has four main modules as task scheduler using ACM-FEF hybrid scheduling algorithm, learning to prediction using ANN for predictions and PSO for learning, learning to optimization using PSO for optimization and ANN for learning, and control mechanism based on inference engine. We proposed two improved variations of PSO named as VB-PSO and R-PSO to be used in prediction and optimization modules in place of PSO. The prediction and optimization module aid the scheduling module in efficient task management by enabling scheduler to make learned decisions.

The learning to prediction mechanism predicts the tasks execution status and machine utilization under given load of the machines/tasks based on history decisions. The variations of PSO are used in learning to prediction mechanism as VB-PSO-NN and R-PSO-NN. In learning to optimization mechanism, an objective function is proposed for enhancing machine utilization and to seek the optimal results based on PSO algorithm. Also, we use the proposed improved variations of PSO (VB-PSO and R-PSO) in the optimization module. We further implement the ANN learning based VB-PSO and R-PSO; where ANN is used to tune the PSO particles' positions for efficiently finding optimal solution.

Our main contributions in this thesis can be listed as

– Scheduling components
    - Learned FEF Scheduling Scheme

    - Hybrid ACM-FEF scheduling scheme
  - Proposal of improved variations of PSO algorithm
    - VB-PSO
    - R-PSO
  - Prediction components
    - PSO learning based ANN weights tuning for enhancing prediction accuracy
  - Optimization components
    - Objective function for maximizing machine utilization
    - ANN learning based PSO parameter tuning for enhancing optimization results

We have developed an emulator of smart factory tasks for simulating real-time task generation and we have evaluated the proposed system based on the simulations using three tasks datasets. The scheduling schemes used for results and comparisons analysis are basic FEF scheduling scheme (FEF without learning factors), predictive FEF scheduling scheme (FEF with ANN learning), learned predictive FEF scheduling scheme (FEF with PSO-NN learning), learned predictive hybrid (ACM-FEF) scheduling scheme, and learned predictive and optimized hybrid scheduling scheme (ACM-FEF with PSO-NN predictive learning and ANN-PSO optimized learning).

The proposed task management mechanism is evaluated based on multiple scenario simulations and performance analysis. We use three task modeling scenarios as candy box factory tasks dataset; user input based simulated tasks dataset and machine cluster tasks dataset. We analyses the results analysis based on the (a) analysis of the prediction module (b) analysis of the optimization module and (c) analysis of the hybrid scheduling scheme. The performances analysis metrics considered are prediction accuracy, tasks instances missing rate, tasks starvation rate, machine utilization rate and machine response time.

The simulations are performed under overloaded tasks load at the machines to examine worst case scenarios. In the simulations, as the tasks grow more than the total machine capacity, some of the tasks' instances are must to drop out.

The results analysis and comparisons clearly show that prediction and optimization modules enhance the machine utilization and scheduler performance. The addition of learning modules further increases the performance by reducing the response times. The PSO based ANN predictions gives higher accuracy and the modification proposed for PSO improve the performance of PSO both in prediction module and optimization module. In performance analysis for candy box factory tasks dataset, we observe that prediction accuracy achieved by PSO-NN is 99.39% in 700 iterations while the prediction accuracy achieved by R-PSO-NN is 99.53 in 700 iterations and the prediction accuracy achieved by VB-PSO-NN is 99.53% in 500 iterations. In performance analysis for simulated tasks dataset, we observe that prediction accuracy achieved by PSO-NN is 99.42% in 800 iterations while the prediction accuracy achieved by R-PSO-NN is 99.69 in 700 iterations and the prediction accuracy achieved by VB-PSO-NN is 99.54% in 200 iterations. In performance analysis for machine cluster tasks dataset, we observe that prediction accuracy achieved by PSO-NN is 98.21% in 800 iterations while the prediction accuracy achieved by R-PSO-NN is 98.42% in 800 iterations and the prediction accuracy achieved by VB-PSO-NN is 98.42% in 300 iterations.

In the comparisons' analysis for candy box factory, we have observed the following improvements. The learned predictive FEF scheduling in comparison to basic FEF scheduling scheme shows an average of 50% reduction in tasks starvation rate and an average of 63.64% reduction in tasks instances missing rate. The learned predictive and optimized hybrid scheduling scheme (proposed task management mechanism) shows an average of 22% increase in machine utilization, and an average of 30% improvement in response times. In the comparisons' analysis for simulated tasks dataset, we have observed the following improvements. The learned predictive FEF scheduling in comparison to basic FEF scheduling scheme shows an average of 77.78% reduction in tasks starvation rate and an average of 78.26% reduction in tasks instances missing rate. The learned predictive and optimized hybrid scheduling scheme (proposed task management mechanism) shows an average of 19.19% increase in machine utilization, and an average of 26.98% improvement in response times. In the comparisons' analysis for machine cluster tasks dataset, we have observed the following improvements. The learned predictive FEF scheduling in comparison to basic FEF scheduling scheme shows an average of 72.23% reduction in tasks starvation rate and an average of 54.17% reduction in tasks instances missing rate. The learned predictive and optimized hybrid scheduling scheme (proposed task management mechanism) shows an average of 27.28% increase in machine utilization, and an average of 36.38% improvement in response times.

Overall, we observe that the learned predictive FEF scheduling in comparison to basic FEF scheduling scheme shows an average of 72.23% reduction in tasks starvation rate and an average of 54.17% reduction in tasks instances missing rate. The learned predictive and optimized hybrid scheduling scheme (proposed task management mechanism) shows an average of 27.28% increase in machine utilization, and an average of 36.38% improvement in response times.

The comparisons analysis shows that proposed task management system, referred as learned predictive and optimized hybrid scheduling scheme in the results analysis, significantly improves the machine utilization rate and drastically drops the tasks instances missing rate and tasks starvation rate. Hence, we can conclude that proposed modules, enhance prediction accuracy, enhance the optimization results and also increase the machine utilization and scheduling results.

References

1) https://www.siemens.com.tr/i/Assets/gida-gunu/170524_KS_Industrie40_Presentation_Siemens_Turkey.pdf

2) What is Industry 4.0? Available at: https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#153613769788

3) http://www.lgcnsblog.com/features/industry-4-0-the-fourth-industrial-revolution-with-it-and-the-manufacturing-industry-sgs-platform-2/#sthash.T1sgnNZk.dpbs

4) Zhong, R.Y., Xu, X. and Wang, L., 2017. IoT-enabled smart factory visibility and traceability using laser-scanners. Procedia Manufacturing, 10, pp.1-14.

5) Nauck, D., et al., Predictive Customer Analytics and Real-Time Business Intelligence, in Service Chain Management, C. Voudouris, D. Lesaint, and G. Owusu, Editors. 2008, Springer Berlin Heidelberg. p. 205-214.

6) Saldivar, A.A.F., Goh, C., Chen, W.N. and Li, Y., 2016, July. Self-organizing tool for smart design with predictive customer needs and wants to realize Industry 4.0. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 5317-5324). IEEE.

7) Lee, J. Smart Factory Systems. Informatik Spektrum 2015, 38, 230–235.

8) Liu, X.F.; Shahriar, M.R.; Al Sunny, S.M.N.; Leu, M.C.; Hu, L. Cyber-physical manufacturing cloud: Architecture, virtualization, communication, and testbed. J. Manuf. Syst. 2017, 43, 352–364

9) 7. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. J. Ind. Inf. Integr. 2017, 6, 1–10.

10) Mowery, D.C.; Rosenberg, N. Technology and the Pursuit of Economic Growth; Cambridge University Press: New York, NY, USA, 1989.

11) J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems, 29 (2013) 1645-1660.

12) R. Y. Zhong, Z. Li, A. L. Y. Pang, Y. Pan, T. Qu, G. Q. Huang, RFID-enabled Real-time Advanced Planning

13) https://www.manufacturing.net/industry40/article/13248934/industry-40-iot-is-key-to-the-smart-factory

14) Lee, J., 2015. Smart Factory Systems. Informatik Spektrum, 38(3), pp.230-235.

15) "Process engineering." [Online]. Available: http://processengineering.theengineer.co.uk/home/contr ol-and-instrumentation/brave-new-world-industry-40- technology/1017121.article.

16) H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the

strategic initiative INDUSTRIE 4.0," 2013.

17) "Bosch." [Online]. Available: http://www.boschsi.com/solutions/manufacturing/industry-4-0/industry4-0.html

18) The Economist Intelligence Unit, "The Internet of Things Business Index: A Quiet Revolution Gathers Pace," 2013.

19) "PRWeb." [Online]. Available: www.prweb.com/releases/2013/12/prweb11430148.ht m.

20) Lopez Research, "Building Smarter Manufacturing With The Internet of Things (IoT)," 2014.

21) Shrouf, F., Ordieres, J. and Miragliotta, G., 2014, December. Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm. In 2014 IEEE international conference on industrial engineering and engineering management (pp. 697-701). IEEE.

22) Baheti, R. and Gill, H., 2011. Cyber-physical systems. The impact of control technology, 12(1), pp.161-166.

23) Krogh, B.H., 2008. Cyber physical systems: the need for new models and design paradigms. Presentation Report.

24) National Institute of Standards and Technology. Workshop report on foundations for innovation in cyber-physical systems, January 2013 <http://www.nist.gov/el/upload/CPS-WorkshopReport-1-30-13-Final.pdf/.

25) Lee, J. and Lapira, E., 2013. Predictive factories: the next transformation. Manufacturing Leadership Journal, 20(1), pp.13-24.

26) Lee, J., Lapira, E., Yang, S. and Kao, A., 2013. Predictive manufacturing system-Trends of next-generation production systems. IFAC Proceedings Volumes, 46(7), pp.150-156.

27) Lee, J., Bagheri, B. and Kao, H.A., 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing letters, 3, pp.18-23.

28) https://cdn.auckland.ac.nz/assets/mech/about/our-research/Industry4/LISMS_poster_Smart%20factory%20modeling.pdf

29) Hermann, M., Pentek, T. and Otto, B., 2016, January. Design principles for industrie 4.0 scenarios. In 2016 49th Hawaii international conference on system sciences (HICSS) (pp. 3928-3937). IEEE.

30) Wang, S., Wan, J., Li, D. and Zhang, C., 2016. Implementing smart factory of industrie 4.0: an outlook. International Journal of Distributed Sensor Networks, 12(1), p.3159805.

31) Sokolov, B. and Ivanov, D., 2015. Integrated scheduling of material flows and information services in industry 4.0 supply networks. IFAC-PapersOnLine, 48(3), pp.1533-1538.

32) Goryachev, A., Kozhevnikov, S., Kolbova, E., Kuznetsov, O., Simonova, E., Skobelev, P., Tsarev, A. and Shepilov, Y., 2013. "Smart Factory": Intelligent System for Workshop Resource Allocation, Scheduling, Optimization and Controlling in Real Time.

In Advanced Materials Research (Vol. 630, pp. 508-513). Trans Tech Publications.

33) Kück, M., Ehm, J., Freitag, M., Frazzon, E.M. and Pimentel, R., 2016. A data-driven simulation-based optimisation approach for adaptive scheduling and control of dynamic manufacturing systems. In Advanced Materials Research (Vol. 1140, pp. 449-456). Trans Tech Publications.

34) Ishii, N. and Talavage, J.J., 1994. A mixed dispatching rule approach in FMS scheduling. International Journal of Flexible Manufacturing Systems, 6(1), pp.69-87.

35) Metan, G., Sabuncuoglu, I. and Pierreval, H., 2010. Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining. International Journal of Production Research, 48(23), pp.6909-6938.

36) Olafsson, S. and Li, X., 2010. Learning effective new single machine dispatching rules from optimal scheduling data. International Journal of Production Economics, 128(1), pp.118-126.

37) Son, Y.J., Rodríguez-Rivera, H. and Wysk, R.A., 1999. A multi-pass simulation-based, real-time scheduling and shop floor control system. Transactions of the Society for Computer Simulation, 16(4), pp.159-172.

38) Priore, P., Gómez, A., Pino, R. and Rosillo, R., 2014. Dynamic scheduling of manufacturing systems using machine learning: An updated review. AI EDAM, 28(1), pp.83-97.

39) Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning internal representations by error propagation", in\Parallel Distributed Processing", DE Rumelhart, JL McClelland eds.

40) Vapnik, V., 2013. The nature of statistical learning theory. Springer science & business media.

41) Quinlan, J.R., 2014. C4. 5: programs for machine learning. Elsevier.

42) Jules, G. and Saadat, M., 2016. Agent cooperation mechanism for decentralized manufacturing scheduling. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(12), pp.3351-3362.

43) Schwiegelshohn, U.; Yahyapour, R. Analysis of first-come-first-serve parallel job scheduling. SODA 1998, 98, 629–638.

44) Arpaci-Dusseau, R.H.; Arpaci-Dusseau, A.C. Operating Systems: Three Easy Pieces; Chapter Scheduling Introduction; ;login: issue: Spring 2017, Vol. 42, No. 1 2014.

45) Ramamritham, K.; Stankovic, J.A. Scheduling algorithms and operating systems support for real-time systems. Proc. IEEE 1994, 82, 55–67.

46) Oh, S.-H.; Yang, S.-M. A modified least-laxity-first scheduling algorithm for real-time tasks. In Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications, Hiroshima, Japan, 27–29 October 1998.

47) Lehoczky, J.; Sha, L.; Ding, Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In Proceedings of the Real-time Systems

Symposium, Santa Monica, CA, USA, 5–7 December 1989; pp. 166–171.

48) Audsley, N.C.; Burns, A.; Richardson, M.F.; Wellings, A.J. Deadline Monotonic Scheduling; University of York, Department of Computer Science: York, UK, 1990.

49) Stankovic, J.A.; et al. Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 460.

50) Chetto, H.; Chetto, M. Some results of the earliest deadline scheduling algorithm. IEEE Trans. Softw. Eng. 1989, 15, 1261.

51) Stewart, D.B.; Khosla, P. Real-time scheduling of sensor-based control systems. IFAC Proc. Vol. 1991, 24, 139–144.

52) Buttazzo, G.C. Rate monotonic vs. EDF: Judgment day. Real-Time Syst. 2005, 29, 5–26.

53) Wu, Y.; Song, X.; Gong, G. Real-time load balancing scheduling algorithm for periodic simulation models. Simul. Model. Pract. Theory 2015, 52, 123–134.

54) Nakahira, Y.; Chen, N.; Chen, L.; Low, S.H. Smoothed Least-laxity-first Algorithm for EV Charging. In Proceedings of the Eighth International Conference on Future Energy Systems, Hong Kong, China, 16–19 May 2017; pp. 242–251.

55) Tabuada, P. Event-triggered real-time scheduling of stabilizing control tasks. Ieee Trans. Autom. Control 2007, 52, 1680–1685.

56) Jejurikar, R.; Gupta, R. Dynamic slack reclamation with procrastination scheduling in real-time embedded systems. In Proceedings of the 42nd annual Design Automation Conference, Anaheim, CA, USA, 13–17 June 2005.

57) Tidwell, T.; Glaubius, R.; Gill, C.; Smart, W.D. Scheduling for reliable execution in autonomic systems. In International Conference on Autonomic and Trusted Computing; Springer: Berlin/Heidelberg, Germany, 2008; pp. 149–161.

58) Dighriri, M.; Alfoudi, A.S.D.; Lee, G.M.; Baker, T.; Pereira, R. Comparison data traffic scheduling techniques for classifying QoS over 5G mobile networks. In Proceedings of the 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, Taiwan, 27–29 March 2017; pp. 492–497.

59) Dighriri, M.; Lee, G.M.; Baker, T. Applying Scheduling Mechanisms Over 5G Cellular Network Packets Traffic. In Third International Congress on Information and Communication Technology; Springer: Singapore, 2019; pp. 119–131.

60) Bala, A.; Chana, I. Autonomic fault tolerant scheduling approach for scientific workflows in Cloud computing. Concurr. Eng. 2015, 23, 27–39.

61) Eker, J.; Hagander, P.; Årzén, Ka. A feedback scheduler for real-time controller tasks. Control Eng. Pract. 2000, 8, 1369–1378.

62) Marzario, L.; Lipari, G.; Balbastre, P.; Crespo, A. Iris: A new reclaiming algorithm for server-based real-time systems. In Proceedings of the Real-Time and Embedded Technology and Applications Symposium, Toronto, ON, Canada, 25–28 May 2004; pp. 211-218.

63) Cho, H.; Ravindran, B.; Jensen, E.D. An optimal real-time scheduling algorithm for multiprocessors. In Proceedings of the Real-Time Systems Symposium, Rio de Janeiro, Brazil, 5–8 December 2006.

64) Buttazzo, G.C.; Bertogna, M.; Yao, G. Limited preemptive scheduling for real-time systems. a survey. Ieee Trans. Ind. Inform. 2013, 9, 3–15.

65) Huang, W.-H.; Chen, Ji.; Zhou, H.; Liu, C. PASS: Priority assignment of real-time tasks with dynamic suspending behavior under fixed-priority scheduling. In Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.

66) Ayele, A.A.; Rao, V.S.; Dileep, K.G.; Bokka, R.K. Combining EDF and LST to enhance the performance of real-time task scheduling. In Proceedings of the International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 18–19 November 2016; pp. 1–6.

67) Kang, Y.S., Park, I.H. and Youm, S., 2016. Performance prediction of a MongoDB-based traceability system in smart factory supply chains. Sensors, 16(12), p.2126.

68) Wu, D., Jennings, C., Terpenny, J., Gao, R.X. and Kumara, S., 2017. A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. Journal of Manufacturing Science and Engineering, 139(7), p.071018.

69) Yan, J., Meng, Y., Lu, L. and Li, L., 2017. Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance. IEEE Access, 5, pp.23484-23491.

70) Hsieh, Y.S., Cheng, F.T., Huang, H.C., Wang, C.R., Wang, S.C. and Yang, H.C., 2012. VM-based baseline predictive maintenance scheme. IEEE Transactions on semiconductor Manufacturing, 26(1), pp.132-144.

71) Chiu, Y.C., Cheng, F.T. and Huang, H.C., 2017. Developing a factory-wide intelligent predictive maintenance system based on Industry 4.0. Journal of the Chinese Institute of Engineers, 40(7), pp.562-571.

72) Wang, J., Zhang, L., Duan, L. and Gao, R.X., 2017. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. Journal of Intelligent Manufacturing, 28(5), pp.1125-1137.

73) Iverson, M.A., Ozguner, F. and Potter, L.C., 1999, April. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. In Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99) (pp. 99-111). IEEE.

74) Kong, X., Lin, C., Jiang, Y., Yan, W. and Chu, X., 2011. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. Journal of network and Computer Applications, 34(4), pp.1068-1077.

75) Li, J., Ma, X., Singh, K., Schulz, M., de Supinski, B.R. and McKee, S.A., 2009, April. Machine learning based online performance prediction for runtime parallelization and task scheduling. In 2009 IEEE International Symposium on Performance Analysis of

Systems and Software (pp. 89-100). IEEE.

76) Wang, L., von Laszewski, G., Huang, F., Dayal, J., Frulani, T. and Fox, G., 2011. Task scheduling with ANN-based temperature prediction in a data center: a simulation-based study. Engineering with Computers, 27(4), pp.381-391.

77) Lee, L.T., Liang, C.H. and Chang, H.Y., 2006, September. An adaptive task scheduling system for Grid Computing. In The Sixth IEEE International Conference on Computer and Information Technology (CIT'06) (pp. 57-57). IEEE.

78) Zhang, D., Liu, Y., Li, J., Xue, C.J., Li, X., Wang, Y. and Yang, H., 2016. Solar power prediction assisted intra-task scheduling for nonvolatile sensor nodes. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(5), pp.724-737.

79) Daly, D.M., Franaszek, P.A. and Lastras-Montano, L.A., International Business Machines Corp, 2012. Prediction based priority scheduling. U.S. Patent 8,185,899.

80) Ghiasi, S., Keller Jr, T.W., Kotla, R. and Rawson III, F.L., International Business Machines Corp, 2008. Scheduling processor voltages and frequencies based on performance prediction and power constraints. U.S. Patent 7,386,739.

81) Jiang, B., Ravindran, B. and Cho, H., 2012. Probability-based prediction and sleep scheduling for energy-efficient target tracking in sensor networks. IEEE Transactions on Mobile Computing, 12(4), pp.735-747.

82) Zhou, J. and Dexter, F., 1998. Method to assist in the scheduling of add-on surgical cases-upper prediction bounds for surgical case durations based on the log-normal distribution. Anesthesiology: The Journal of the American Society of Anesthesiologists, 89(5), pp.1228-1232.

83) Zeng, F., Zhang, R., Cheng, X. and Yang, L., 2017. Channel prediction based scheduling for data dissemination in VANETs. IEEE Communications Letters, 21(6), pp.1409-1412.

84) Israr Ullah, PHD Thesis "Enhanced Optimization Scheme based on Learning for Efficient Energy Consumption in Smart Greenhouse", Jeju National University, December, 2018.

85) Shao, G., Shin, S.J. and Jain, S., 2014, December. Data analytics using simulation for smart manufacturing. In Proceedings of the Winter Simulation Conference 2014 (pp. 2192-2203). IEEE.

86) Qian, F., Zhong, W. and Du, W., 2017. Fundamental theories and key technologies for smart and optimal manufacturing in the process industry. Engineering, 3(2), pp.154-160.

87) Gjeldum, N., Mladineo, M., Crnjac, M., Veza, I. and Aljinovic, A., 2018. Performance analysis of the RFID system for optimal design of the intelligent assembly line in the learning factory. Procedia Manufacturing, 23, pp.63-68.

88) Chen, T. and Lin, Y.C., 2017. Feasibility evaluation and optimization of a smart manufacturing system based on 3D printing: A review. International Journal of Intelligent Systems, 32(4), pp.394-413.

89) Edgar, T.F. and Pistikopoulos, E.N., 2018. Smart manufacturing and energy systems. Computers & Chemical Engineering, 114, pp.130-144.

90) C. Fang, X. Liu, P. M. Pardalos, and J. Pei, “Optimization for a three-stage production system in the Internet of Things: procurement, production and product recovery, and acquisition,” Int. J. Adv. Manuf. Technol., vol. 83, no. 5–8, pp. 689–710, 2016.

91) K. Kang, T. Qu, D. X. Nie, T. Zhang, Z. Z. Wang, and G. Q. Huang, “Production system multi-stage synchronization based on collaborative optimization under the Internet-of-Things environment,” in Networking, Sensing, and Control (ICNSC), 2016 IEEE 13th International Conference on, 2016, pp. 1–6.

92) Y. Zhang and F. Tao, Optimization of Manufacturing Systems Using the Internet of Things. Academic Press, 2016.

93) P. Ghashghaee, “Smart manufacturing: role of Internet of Things in process optimization,” 2016.

94) X. Liu, J. Pei, L. Liu, H. Cheng, M. Zhou, and P. M. Pardalos, Optimization and Management in Manufacturing Engineering: Resource Collaborative Optimization and Management Through the Internet of Things, vol. 126. Springer, 2017.

95) X.-S. Yang, S. Deb, and S. Fong, “Accelerated particle swarm optimization and support vector machine for business optimization and applications,” in International Conference on Networked Digital Technologies, 2011, pp. 53–66.

96) Chen K (2012) Procurement strategy and coordination mechanism of the supply chain with one manufacturer and multiple suppliers. Int J Prod Econ 138(1):125–135

97) Mukhopadhyay SK, Ma H (2009) Joint procurement and production decisions in remanufacturing under quality and demand uncertainty. Int J Prod Econ 120(1):5–17

98) Fu Q, Lee CY, Teo CP (2010) Procurement management using option contracts: random spot price and the portfolio effect. IIE Trans 43(11):793–811

99) Xu H (2010) Managing production and procurement through option contracts in supply chains with random yield. Int J Prod Econ 126(2):306–131

100) Kim SH, Netessine S (2013) Collaborative cost reduction and component procurement under information asymmetry. Manag Sci 59(1):189–206

101) Hu F, Lim CC, Lu Z (2014) Optimal production and procurement decisions in a supply chain with an option contract and partial backordering under uncertainties. Appl Math Comput 232(1): 1225–1234

102) Galbreth MR, Blackburn JD (2006) Optimal acquisition and sorting policies for remanufacturing. Prod Oper Manag 15(3):384–392

103) Teunter RH, Flapper SDP (2011) Optimal core acquisition and remanufacturing policies under uncertain core quality fractions. Eur J Oper Res 210(2):241–248

104) Zhou SX, Yu Y (2011) Optimal product acquisition, pricing, and inventory management for systems with remanufacturing. Oper Res 59(2):514–521

105) Zeng AZ (2013) Coordination mechanisms for a three-stage reverse supply chain to increase profitable returns. Nav Res Logist 60(1): 32–45

106) Konstantaras I, Skouri K (2010) Lot sizing for a single product recovery system with variable setup numbers. Eur J Oper Res 203(2):326–335

107) Kim T, Goyal SK (2011) Determination of the optimal production policy and product recovery policy: the impacts of sales margin of recovered product. Int J Prod Res 49(9):2535–2550

108) Kumar M, Husian M, Upreti N, Gupta D (2010) Genetic algorithm: review and application. Int J Inform Tech Knowl Manag 2(2):451– 454

109) Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344:243–278

110) Suman B, Kumar P (2006) A survey of simulated annealing as a tool for single and multiobjective optimization. J Oper Res Soc 57: 1143–1160

111) Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridization, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124

112) McKay, K.N. and Wiers, V.C., 2003. Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. Computers in Industry, 50(1), pp.5-14.

113) Wan, J., Chen, B., Wang, S., Xia, M., Li, D. and Liu, C., 2018. Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Transactions on Industrial Informatics, 14(10), pp.4548-4556.

114) Yin, L., Luo, J. and Luo, H., 2018. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. IEEE Transactions on Industrial Informatics, 14(10), pp.4712-4721.

115) Kang, Y.S., Park, I.H. and Youm, S., 2016. Performance prediction of a MongoDB-based traceability system in smart factory supply chains. Sensors, 16(12), p.2126.

116) Wu, D., Jennings, C., Terpenny, J., Gao, R.X. and Kumara, S., 2017. A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. Journal of Manufacturing Science and Engineering, 139(7), p.071018.

117) Yan, J., Meng, Y., Lu, L. and Li, L., 2017. Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance. IEEE Access, 5, pp.23484-23491.

118) Chiu, Y.C., Cheng, F.T. and Huang, H.C., 2017. Developing a factory-wide intelligent predictive maintenance system based on Industry 4.0. Journal of the Chinese Institute of Engineers, 40(7), pp.562-571.

119) Wang, J., Zhang, L., Duan, L. and Gao, R.X., 2017. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. Journal of Intelligent Manufacturing, 28(5), pp.1125-1137.

120) Aversa, R., Petrescu, R.V., Petrescu, F.I. and Apicella, A., 2016. Smart-factory: Optimization and process control of composite centrifuged pipes. American Journal of

Applied Sciences, 13(11), pp.1330-1341.

121) Qian, F., Zhong, W. and Du, W., 2017. Fundamental theories and key technologies for smart and optimal manufacturing in the process industry. Engineering, 3(2), pp.154-160.

122) Qian, F., Zhong, W., Du, W., Wang, H., Yan, X., Lv, Z., Jiang, Q., Chen, X., Song, B., Ma, Y. and Shi, H., 2017. Smart and optimal manufacturing: The key for the transformation and development of the process industry. Engineering, 3(2), p.151.

123) Jang, H.S., 2019, May. A Study on Optimal Automation Level of the Smart Factory for Competitive Display Manufacturing. In 2019 년 한국산업경영시스템학회 춘계학술대회 (pp. 191-193).

124) Gjeldum, N., Mladineo, M., Crnjac, M., Veza, I. and Aljinovic, A., 2018. Performance analysis of the RFID system for optimal design of the intelligent assembly line in the learning factory. Procedia Manufacturing, 23, pp.63-68.

125) Reinfurt, L., Falkenthal, M., Breitenbücher, U. and Leymann, F., 2017. Applying IoT Patterns to Smart Factory Systems. Advanced Summer School on Service Oriented Computing, Summer SOC.

126) Zawadzki, P. and Żywicki, K., 2016. Smart product design and production control for effective mass customization in the Industry 4.0 concept. Management and Production Engineering Review, 7(3), pp.105-112.

127) Prinz, C., Morlock, F., Freith, S., Kreggenfeld, N., Kreimeier, D. and Kuhlenkötter, B., 2016. Learning factory modules for smart factories in industrie 4.0. Procedia CiRp, 54, pp.113-118.

128) Zheng, P., Sang, Z., Zhong, R.Y., Liu, Y., Liu, C., Mubarok, K., Yu, S. and Xu, X., 2018. Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives. Frontiers of Mechanical Engineering, 13(2), pp.137-150.

129) McCulloch, W.; Walter, P. A Logical Calculus of Ideas Immanent in Nervous Activity. Bull. Math. Biophys. 1943, 5, 115–133.

130) Minsky, M.; Papert, S. Perceptrons: An Introduction to Computational Geometry; MIT Press: Cambridge, MA, USA, 1969.

131) Artificial Neural Networks as Models of Neural Information Processing|Frontiers Research Topic. Available online: https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing (accessed on 30 March 2018).

132) Artificial Neuron Output. Available online: https://en.wikipedia.org/wiki/Artificial_neuron (accessed on 4 May 2018).

133) Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Western Australia, 27 November–1 December 1995; pp. 1942–1948.

134) Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. Swarm Intell. 2007, 1, 33–57.

135) Jules, G. and Saadat, M., 2016. Agent cooperation mechanism for decentralized manufacturing scheduling. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(12), pp.3351-3362.

136) Malik, S., Ahmad, S., Ullah, I., Park, D.H. and Kim, D., 2019. An Adaptive Emergency First Intelligent Scheduling Algorithm for Efficient Task Management and Scheduling in Hybrid of Hard Real-Time and Soft Real-Time Embedded IoT Systems. Sustainability, 11(8), p.2192.

137) Malik, S. and Kim, D., 2018. Prediction-learning algorithm for efficient energy consumption in smart buildings based on particle regeneration and velocity boost in particle swarm optimization neural networks. Energies, 11(5), p.1289.

138) Malik, S., Ahmad, S., Kim, B.W., Park, D.H. and Kim, D., 2019. Hybrid Inference Based Scheduling Mechanism for Efficient Real Time Task and Resource Management in Smart Cars for Safe Driving. Electronics, 8(3), p.344.

139) Chen, B., Wan, J., Shu, L., Li, P., Mukherjee, M. and Yin, B., 2017. Smart factory of industry 4.0: Key technologies, application case, and challenges. IEEE Access, 6, pp.6505-6519

140) http://bry-air.com/casestudies/dehumidication-candy-processing-drying-packaging/