IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 3, 2022, accepted April 30, 2022, date of publication May 10, 2022, date of current version May 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3173157

A Criticality-Aware Dynamic Task Scheduling Mechanism for Efficient Resource Load Balancing in Constrained Smart Manufacturing Environment

ANAM NAWAZ KHAN[1], NAEEM IQBAL[1], ATIF RIZWAN[1], SEHRISH MALIK[2], RASHID AHMAD[3,4], AND DO HYEUN KIM[1,5]

[1]Department of Computer Engineering, Jeju National University, Jeju-si 63243, Jeju Special Self-Governing Province, Republic of Korea
[2]Data Driven Software Engineering Department, Simula Research Laboratory, 0164 Oslo, Norway
[3]Big Data Research Center, Jeju National University, Jeju 63243, Republic of Korea
[4]Department of Computer Science, COMSATS University Islamabad, Attock Campus, Attock 43600, Pakistan
[5]Department of Advanced Technology, Research Institute, Jeju National University, Jeju 63243, Republic of Korea

Corresponding author: Do Hyeun Kim (kimdh@jejunu.ac.kr)

This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-00188, Open-source development and standardization for AI enabled IoT platforms and interworking). Any correspondence related to this paper should be addressed to Do Hyeun Kim.

ABSTRACT Smart factory is an exemplification of the Industrial Internet of Things (IIoT), connecting devices to expedite the production process and delivery of customized products. Today's intelligent manufacturing systems strive to develop low cost product with robust manufacturing process to compete the global market challenges. Though the quest to provide robust, reliable, adaptive, proactive, and real-time services to smart factory production processes has got little attention. To this aim, establishment of a robust solution for efficient resource utilization, load balancing and task scheduling has become a de-facto necessity. This paper presents an enhanced learning assisted task scheduling mechanism based on task Criticality and Collapse Aware Scheduling (CCAS) algorithm. The proposed mechanism is developed using two modules; namely task scheduling mechanism based on task criticality and collapse aware strategy, and an ensemble prediction model i.e., Gradient Boosting Decision Tree (GBDT) to proactively predict the machine utilization and task safe execution status. The proposed ensemble learning framework provides high level feature abstraction by learning the task parameters to predict task status and machine utilization. Furthermore, an intelligent scheduling mechanism is developed for optimal resource allocation to maximize the production in constrained smart manufacturing machine's network. Extensive experimentation and comparative analysis with the conventional Rate Monotonic (RM) algorithm has been carried out to validate the performance of the proposed approach. The results demonstrate that the proposed enhanced scheduling mechanism yields superior performance in terms of response time, task dropout and starvation rates compared to the conventional RM method. The developed predictive CCAS scheduling reduced response time, task dropout and starvation rate by 13%, 15%, and 14% respectively, compared to the baseline RM scheduling approach. The results shows that the CCAS shall enhance the resource utilization in smart factory yielding enhanced productivity and reduced cost of production.

INDEX TERMS Industrial Internet of Things, smart manufacturing, embedded IoT, real-time task scheduling.

I. INTRODUCTION

Industry 4.0 has emerged as a revolutionary paradigm that focuses on intelligent factories and smart production processes involving the Internet of Things (IoT) and the Cyber-Physical Systems (CPS). Manufacturing industries are undergoing the fourth industrial revolution, also termed Industry 4.0 or I4.0, involving integration of smart factory physical and digital systems [1]. Due to advancements in technology and rapid growth of global economies in

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed M. Elmisery.

the recent past,industries globally experienced a remarkable change due to international market intensified competition level. Current situation require changes in the manufacturing process with faster reconfiguration and higher agility [2]. The smart factory mainly encompasses the IoT that hold smart machines, intelligent automation networked sensors, and CPS enabled intelligent robots aspiring to achieve digitalization in the production process. A detailed analysis and review of IoT sensor networks, big data analytics and its role in sustainable manufacturing is presented by [3]. IoT and CPS facilitate communication between humans and machines connected via a network [4]. Presently, smart factories are no longer confined to encapsulated hierarchy-based physical and logical systems; they are transformed into heterogeneous systems accompanied by loosely coupled CPS communicating through a unified network. Today, IoT devices generate massive amounts of data. Data availability has created opportunities for researchers to find patterns in data and discover knowledge that facilitates production optimization. Existing solutions mainly devised for optimal smart factory task scheduling issues are established upon two approaches known as Precise and Exact. Precise methods obtain optimal global solutions by searching the whole solution space. Despite that, such methods are not efficient at solving large-scale scheduling problems. Although, approximate methods are computationally inexpensive, they still do not guarantee optimal solutions [5]. An alternate solution to smart factory scheduling is based on a dynamic scheduling strategy that guarantees a reliable and flexible solution. Dynamic scheduling solutions encompasses learning-based approaches [6], agent-based approaches [7], heuristic methods [8], evolutionary computing based solutions [9] and variable neighbourhood methods [10]. Machine learning (ML) and Artificial Intelligence (AI) [11] techniques are widely adopted in many new-generation manufacturing systems for accurate decision-making. At the moment, ML and AI techniques have attracted enormous attention from the research community for provisioning solutions to optimize resources usage, reduce maintenance cost, fault detection, predictive maintenance, to name a few [12]. Machine learning ensemble learning framework [13] are widely adapted to improve the manufacturing process due to features such as flexibility, improved performance and data insensitivity [14]. Ensemble learning frameworks are classified into two types, homogeneous and heterogeneous ensemble modelling [15]. As the name suggests homogeneous ensembles consists of single type of base models or learning algorithm that utilizes same feature selection method build upon different data subsets. For instance, popular homogeneous ensemble methods include bagging and boosting approaches [16] that are capable of generating diversity through random sampling where all base learner possess equal weight. On the other hand, heterogeneous modeling involves different learning algorithms such as stacking [17]. However, the present work implements GBDT that involves fitting a decision tree at each iteration in a direction of error minimization. Today's intelligent manufacturing systems strive to develop innovation, low cost, and highly efficient products [18]. Correspondingly, smart factories need to meet the global market challenges and suspire robust, and flexible production processes. Short manufacturing cycles, lower operating costs, real-time exception handling, and optimal control have become de-facto necessities for smart production and innovative factory solutions. As the complex smart factory systems involve multi-constraints and challenges to devise solutions for optimizing intelligent factory resources [19]. Furthermore, smart production systems are subjected to random critical event-driven and periodic tasks. Therefore, a scheduling solution is substantial for efficiently allocating resources to machines to execute tasks optimally [20]. Previously a limited number of studies focused on devising task scheduling systems based on consideration of complexity associated with practical scheduling systems [21]. However, such systems fail to deliver reliable performance; for instance, short sensing intervals cause the scheduler to be flooded with periodic sensing data added with massive critical and non-critical event-driven tasks. Existing solution devised for improving the productivity and efficiency of smart production processes require efficient task scheduling. Unfortunately, the developed solutions are unable to exploit the heterogeneous nature of real-time tasks for maximizing the resource utilization in resource constrained environments. Existing scheduling algorithms such as Shortest Job First (SJF), First Come First Serve (FCFS), Sort and Fit (SAF) and OctaSort (OS) often fail to meet the task deadlines in-case of varying task types and resources [22]. Despite all developments,finding an optimal task scheduling strategy for smart factory is a ever fallowed. To this aim our proposed approach provides a learning-assisted task scheduling solution for dynamic scheduling of production tasks in constrained environments. The proposed scheduling approach is adaptive to various types of tasks, jobs, processes, machines, and network requirements. The proposed scheduling algorithm encompasses task criticality and task collapse aware strategy for fair allocation of resources. Furthermore, an ensemble learning framework is developed to assist the scheduling model in making informed task scheduling decisions by providing task execution and machine utilization status to improve production process in a smart factory. The proposed system is compared with the classic RM scheduling algorithm for performance evaluation based on response time, machine utilization, task starvation, dropout rate, and latency. The core contributions of the proposed learning to prediction task scheduling mechanism are stated as follows.

- Development of an enhanced task scheduling mechanism based on task criticality and collapse aware scheduling scheme for efficient resource load-balancing and task management in smart factory.
- Consideration of task heterogeneity and priorities to maximize resource utilization in constrained smart manufacturing environment through introduction of Crticality First Measure (CFM) and Collapse Measure (CM).

- Development of a learning assisted task scheduling scheme using GBDT model to enhance the performance of the scheduling mechanism through predictive analytics.
- Generation and execution of control commands based on inference rule engine for optimal actuator control.

The rest of the paper is organized as follows: Section II presents the state of art in field of smart factory task scheduling, resource utilization and load balancing solutions, in addition some preliminaries to task scheduling in smart factory and current challenges are also highlighted. Section III presents the operational overview of predictive task scheduling mechanism along with detailed description of the proposed scheduling algorithm. Section IV presents a detailed description of task modeling and simulation setup for the developed system. presents experimental setup of the proposed enhanced predictive scheduling mechanism. Section V shows the experimental results analysis. Section VI provides a conclusion of the proposed enhanced task scheduling mechanism.

II. RELATED WORK

This section presents a review of existing techniques applied to the domain of task scheduling moreover limitations of developed systems are elaborated. Task scheduling involves making decisions about jobs/ tasks ordering, resource allocation [23], and task execution based on the number of tasks waiting in the machine queue to be executed [24]. Afterward, the routing and execution of tasks are done [25]. Each task has an associated task id, deadline, task type, and other parameters. The employed scheduling algorithm (preemptive or non-preemptive) provides a scheduling policy to schedule various task types (periodic or event-driven). Event-driven tasks are generally flexible, with varying arrival times than periodic tasks with regular arrival times [26]. Smart production environment can have various task types based on their nature and characteristics. The tasks can be either periodic, non-periodic or event-driven. Event-driven tasks are generally flexible, having irregular arrival times comparative to periodic tasks that posses regular arrival time [27].

The literature on scheduling comprises of four methods for scheduling problems that are exact methods, heuristic approaches, simulation based approaches and AI based approaches [28]. Exact methods provide optimal solution through methods like Dynamic Programming (DP), Integer Programming (IP), Branch and Bound (BB) to name a few. However this approach is limited because of its applicability only to small size scheduling problems. Contrarily for large size (NP complete) problem the time complexity of solution approaches become manifolds [29]. Task scheduling for the cloud-based system is emerging research for effective resource utilization in the cloud environment. Previously many researchers attempted to devise a solution based on scheduling algorithms [30], [31]. Recently ML [32] and AI models [33] have been successfully applied to task scheduling through real-time intelligent decision making [27]. ML approaches have also outperformed conventional methods in complex areas such as the cloud. Search Space Exploration, Mixed Integer Programming (MIP), and Reinforcement Learning (RL) provided convincing results to deal with a complex cloud environment. [31] proposed a task scheduling for cloud applications by minimizing the computational cost through efficient resource utilization. An adaptive task scheduling mechanism based on Netflow prediction and clustering is proposed in [33] to overcome the stability and load balancing in cloud environments. A Q-learning-based task scheduling solution for optimal resource utilization and scheduling in cloud environment is presented in [34]. Similar works has been proposed by [35] to provide machine learning based solutions for sustainable smart spaces. Deep Learning (DL) frameworks are also valuable for providing predictive analytics for optimal task placement. However making accurate forecast about various aspects of the production process is a challenging task due to dynamism and heterogeneity of real-time tasks. Time-series prediction based on Deep Convolution Neural Networks (Deep-CNN) has been proposed by [36]. Another DL solution has been developed by [37] to improve the process performance in smart production systems. A DL based prediction model is developed by [38] for machine speed prediction to optimize the production process. A task scheduling algorithm based on deep Q-nets (DQN) has been proposed by [39] to determine the scheduling order of the tasks. A novel energy-efficient task scheduling mechanism is developed for real-time scheduling tasks. The considered scheduling architecture can deal with various task types [40]. An uncertainty-aware scheduling algorithm is presented in [41] for dynamically scheduling workflows in the cloud environment. The result of the experiments shows an improved task starvation rate. ML-based scheduling solutions support adaptive scheduling through learning by running simulations and building a knowledge base. In [42] authors provided a detailed review of ML applications in task scheduling. In [43] authors presented a dynamic scheduling strategy for smart factory production management. [44] Proposed a learning-aided dynamic scheduling framework that involves a knowledge base for selecting rules. The authors proposed an efficient and energy-aware scheduling mechanism for load balancing [45]. In article [30] proposed a reinforcement learning-based solution for a manufacturing system. In [46], the authors proposed a dynamic feature selection method for semi-conductor manufacturing based on Genetic Algorithm (GA) and NN. A notable work proposed an AI based data driven solution for smart manufacturing of semi-conductor material [47]. Prediction of manufacturing outcomes is an essential aspect of task scheduling because of the high production cost. Resource wastage can be effectively mitigated through accurate and timely smart factory production parameters predictions. Learning from historical data in real-time enables smart factory scheduling to be more responsive, predictive, and proactive. Predictive analytics has been

part of smart factory optimization process for a long. In [48], the authors claimed that ensemble learning approaches could efficiently resolve the complexities of smart factories by predicting parameters that prevent failures beforehand. In [49], the authors employed a prediction mechanism to predict the health of smart factory tools. ANN, Support Vector Machines (SVM), and Random Forest (RF) are widely used to predict tool wear in smart factories. Many researchers developed innovative manufacturing scheduling solutions based on centralized and hierarchical systems for providing global schedules for manufacturing systems; however, they are not efficient at handling disturbances. Therefore developing an integrated, flexible, and robust scheduling mechanism is the need of hour [50]. Task scheduling is challenging for various manufacturing industries due to resource constraints and conflicting objectives [51]. Efficient task scheduling plays a central role in enhancing the productivity and efficiency of smart manufacturing systems. Tasks scheduling involves allocating resources (limited or shared) to various tasks, mainly sensing and actuating tasks, production jobs, and processes to be executed within specified time constraints for achieving maximum efficiency and the minimum cost of production [52]. In case of exception, manufacturing processes require feedback from control modules for actuation accordingly [53]. As constrained manufacturing systems have limited resources therefore, an optimal task scheduling must is highly substantial [54]. Despite of advancement in research and technology the existing solutions does not guarantee optimal solutions and their performance is highly influenced by the criteria for optimization, task load, and configuration of system [55]. A system that can optimally allocate resources, with efficient resources load balance through self-learning and prediction, minimize task starvation and sprucely deal with unexpected scenarios.

III. PROPOSED TASK SCHEDULING MECHANISM BASED ON TASK CRITICALITY AND COLLAPSE AWARE SCHEDULING STRATEGY

This section presents the proposed methodology for real time task scheduling mechanism. The propose scheduling algorithm is named as CCAS algorithm. The proposed work aims to schedule tasks with varying task natures, priorities, deadline and constraints in overloaded system scenarios. Smart production IoT systems encompasses periodic and event driven tasks. Periodic tasks involve collecting environment sensing data after every few seconds through installed sensors in the smart spaces. Event driven tasks occurs in response to an abnormal event and possess high priority. The task parameters and their detailed description is described in Table 1. The tasks are categorized into four types for understanding the task nature and context for awareness of associated dependencies so that tasks must be scheduled correspondingly. The proposed system prevents missed deadlines and task starvation when the scheduler gets flooded with periodic sensing data added with massive critical and non-critical event-driven tasks. The scheduling mechanism considers two basic tasks types and two sub task types. Basic task types cover Event Driven (ED) tasks and Periodic tasks (PT). The ED task types are further categorized as critical Event Driven (CED) and Non-Critical Event Driven (NCED) while periodic tasks are sub divided into two types Priority Periodic (PP) and Non-Priority Periodic (NPP). The detailed description of each task type is as follows, in case of non-critical event driven task type the tasks must be executed any time before deadline. Algorithm 1 presents the detailed working of CCAS mechanism.

The ED tasks occur due to reaction to an event or because of unusual system behavior and are assigned the highest priority since they need to be executed right away. The division of task types into sub types help system achieve more flexibility.

Algorithm 1 An Enhanced CCAS Algorithm for Constrained Smart Manufacturing Systems

Data: Tasks in queue $task\_queue = T_1, T_2, \ldots, T_n$
Result: Optimal tasks execution
initialization;
$T_u = \frac{ET}{TP}$
// Compute Task Utilization
$U_{bound} = t_n \times (2^{1/t_n} - 1)$
// Compute upper bound
if $T_U \leq t_n \times (2^{1/t_n} - 1)$ **then**
  // Selection of best machine as per task type
  $Task \leftarrow get\_optimal\_task(task\_queue)$
  $Execute\_Task(Task)$
Function get_optimal_task(*queue*)**:**
  for *task in queue* **do**
    if *task.type == CED* **then**
      $Task \leftarrow CED(closest - deadline)$
    else if *task.type == NCED AND PPT not in queue* **then**
      $Task \leftarrow NCED(closest - deadline)$
      if *task.type == NCED AND PPT in queue* **then**
        if *task.type==PPT* **then**
          $Task \leftarrow PPT(closest - deadline)$
        Calculate CFM $\leftarrow$ *Criticality First Measure*
        $CFM(t) \leftarrow Slack$
    if *CFM == 0* **then**
      $Task \leftarrow PPT(closest - deadline)$
      else
        $Task \leftarrow ED(closest - deadline)$
      else if *task.type == PPT AND NPPT* **then**
        Calculate CM $\leftarrow$ *Collapse Measure*
      if *CM == 0* **then**
        $Task \leftarrow NPP(starving - task)$
      else
        $Task \leftarrow PPT$
    return Task

TABLE 1. Task features and description.

	Task Notations	Description
1	T_{id},	It represents unique identifier corresponding to each Task,
2	T_{at}	It represents Task arrival time at system.
3	T_{et}	It represents Time required to complete execution.
4	T_{st}	It represents Tasks start time at machine.
5	T_{ft}	It represents Tasks finish time at machine.
6	T_{pr}	It represents Tasks priority.
7	T_p	It represents task period.
8	V_{on}	It represent virtual object name x at time t.
9	J_n	number of jobs needed by the task
10	P_b	It represents the value of priority (0/1)
11	C_{bit}	It represents the criticality level for event driven task set
12	M_{id}	It represents unique Machine identifier
13	J_{id}	It represents unique job identifier

CED tasks cannot afford to wait for execution in the queue as compared to NCED tasks. NCED tasks must also be executed before deadline. The ED tasks are recognized by the system using Criticality Bit (CB), If CB=0 then the task is NCED else it is a CED task. In contrast PT arrive regularly and periodically at the system. PP tasks possess a priority with respect to their task period and must be executed before their deadlines. Priority periodic tasks are prioritized over non priority periodic tasks. The periodic tasks are identified using Priority Bit (PB). PB=1 make the system aware that the task is a PP task and is able to preempt the higher priority task if being starved. PB=0 is an indicator of NPP tasks that must not preempt any high priority tasks. If a task possesses high urgency or priority then it must be timely executed. A task model can be explained by the Equation 1. The task model involves task ID, machine ID, arrival time, priority, execution time, finish time, start time, job id, number of jobs, Priority bit for periodic tasks and criticality bit for event driven tasks.

$$\text{Task}_{\text{profile}} = \{T_{id}, M_{id}, T_{pr}, T_{at}, T_{et}, T_{ft}, T_{st}, J_{id}, J_n, V_{on}, P_{bit}, C_{bit}\} \quad (1)$$

CCAS follows a customized scheduling policy for smart factory task scheduling and loads balancing. Fig. 1 presents a detailed flowchart of the proposed CCAS scheduling approach. The proposed efficient resource load balance RM scheduling maintains a task queue for all incoming tasks. The scheduling policy introduces a global scheduler whose job is to manage the load with efficient load balancing strategies by assigning tasks to the processing machines such that resources are handled efficiently. After task arrival the global scheduler manages the task queue and calculate the task utilization (T_u) based on the execution time and task period. Equation 2 presents the formula of computing task utilization.

$$T_u = ET/TP \quad (2)$$

Afterwards, the upper bound is calculated as per RM scheduling. Upper bound represent the schedulability analysis. For all task in a task queue the least upper bound is represented by the following Equation 3.

$$U_{\text{bound}} = t_n \times \left(2^{1/t_n} - 1\right) \quad (3)$$

For any task it is necessary to undergo feasibility check first, if task utilization is less than the least upper bound only then a processor is selected and task is assigned to that processor. The priority of a task is inversely proportional to task period and the assignment of priorities to tasks decides how and in what manner the tasks will be allocated to the machines. Along this strategy an efficient and enhanced rate- monotonic resource allocation scheduling policy is implemented such that task are assigned priorities based on the deadline of the task. Tasks having closer deadlines are given priority for execution on selected machines. Moreover, we introduced two learning factors namely CFM and CM that signifies the execution of emergency and critical tasks and CM that provides the feasibility of executing high priority tasks taking account of starved tasks if any.

A. CRITICALITY FIRST MEASURE

The aforementioned learning components are computed during real-time scheduling using prediction model to help scheduler efficiently allocate tasks to machines and avoid load unbalancing. CFM ascertains whether the system should execute a NCED task or a PP task. CFM is computed between NCED and PP task. By criticality the urgency of the task is quantified so that the task must be executed right away. For that firstly the scheduler checks the criticality tag provided by the CB. If the task is a CED it is executed right away. While if no CED arrived at system, the proposed scheduling mechanism computes Slack for the available tasks is computed for NPPT and NCED tasks. Provided any task, slack indicates the difference between execution time required by a task and deadline of the task. Equation 4 describes the slack calculation to check either NCED task should be executed first or the starving NPP task.

$$S_t(NCED) \geq y \times (S_t(NPP)) \quad (4)$$

The equation is used for slack computation for NCED and NPP task and the distance between both slacks should be y times; while y is initialized with the value 3 and learned gradually using prediction model as more tasks arrives at the system and with the passage of time history is created. The prediction model predicts the value of y; that gives a safe distance between NCED tasks and non-priority periodic tasks.

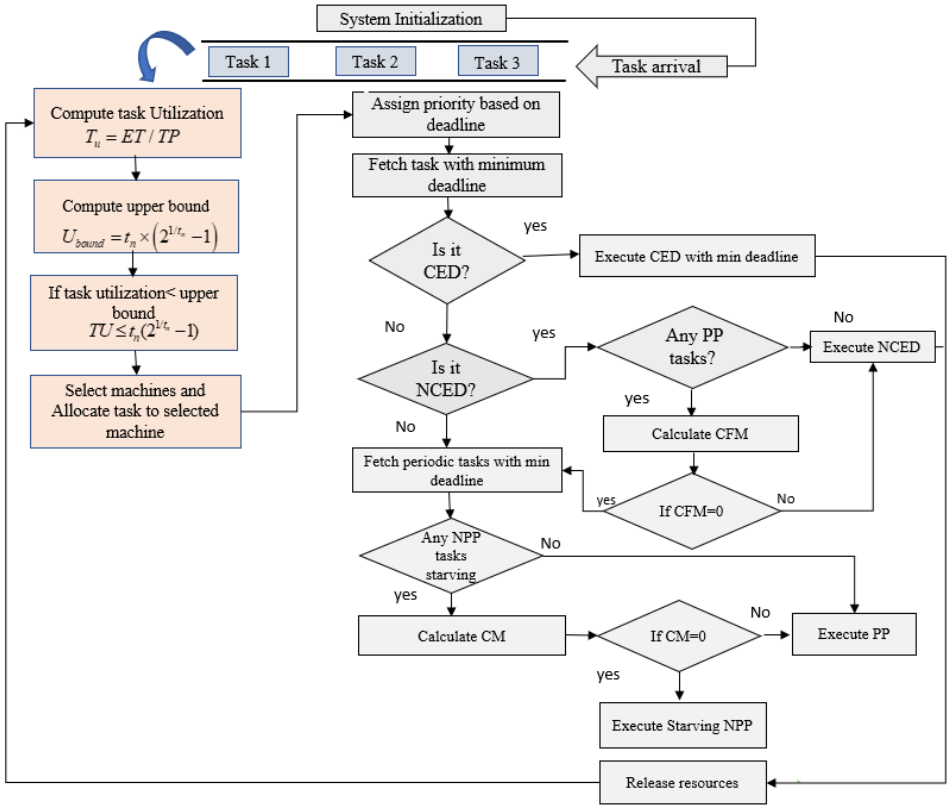

FIGURE 1. Flow model of task CCAS scheme for constrained manufacturing environment.

The process begins when a scheduler based on task parameters, schedule and execute tasks, concurrently a history is also being maintained with the following information, including time-stamp, number of tasks at current time-stamp, task type, priority, preemption threshold, Slack and task drop-out rate Fig. 2.

The log maintained by the system helps scheduler intelligently take decisions by learning from past decisions. This facilitates intelligent decision making as the system grows it learns from historical logs based upon the value of y in Equation 4. If above equation is true CFM is set to 0, task is categorized as not critical and non-urgent thus making a room for starving task having low priority to be executed, else it is set as CFM=1 suggesting that the task type is NCED task and should be executed first. In this way fair allocation of resources is ensured. If NCED and NPPT are simultaneously present, Criticality First Measure (CFM) is computed to make sure the starving task must execute first. CFM quantifies the criticality level of tasks that distinguishes it from other tasks. CFM is a tag for scheduler to identify the task to be executed right away. Correspondingly the scheduler schedules the tasks based on preset priorities with in specified deadlines and try to minimize task starvation for tasks having lower priorities. The proposed scheduling framework aids faster execution of ED tasks by prioritizing them over periodic task while CED tasks are preferred over NCED. In the similar manner priority periodic tasks are preferred over non-priority periodic tasks in case of Periodic tasks.

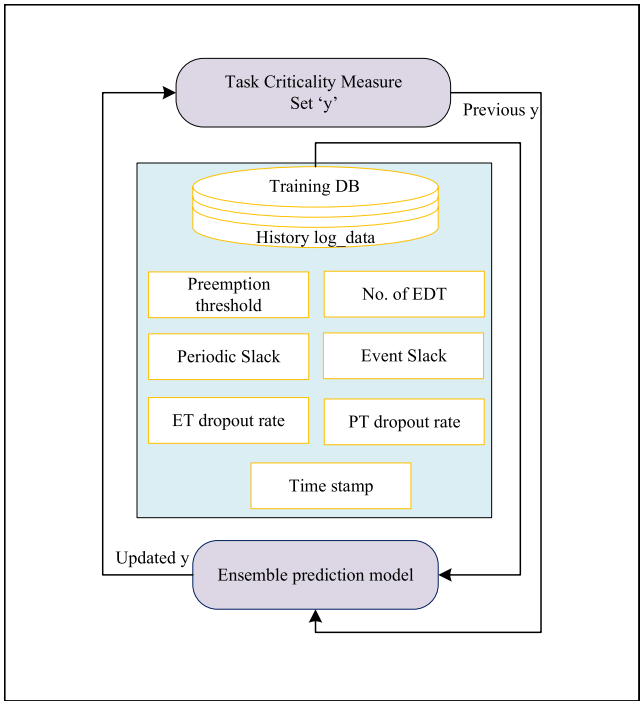

FIGURE 2. Task criticality computation mechanism.

B. COLLAPSE MEASURE

In case of available periodic task, CM is calculated to decide whether starving NPP task can be safely executed or to execute PP tasks. The collapse measure is responsible to

Algorithm 2 An Ensemble (GBDT) Prediction Model for Learning-Assisted Scheduling

Data: Task Data-set:
Taskparameters $[(a_1, b_1), (a_2, b_2) \ldots (a_n, b_n)]$
Resampling ratio (θ);
No. of training rounds (R);
Result: Predict Task Execution Status and Machine Processing Capacity Utilization
foreach *time-stamp* **do**
  Apply bootstrap to acquire training data-set having size M= θ
  M= $(a_1, b_1), (a_2, b_2) \ldots (a_n, b_n)$
  Initialize $\leftarrow f_0^t(a) = \arg\min_{\gamma} \sum_{i=1}^{M} L(b_i, \gamma)$;
  for $p = 1,2,3 \ldots P$; **do**
    for $i = 1,2,3 \ldots M$ *compute* **do**
      $r_{mi} = -\left[\frac{\partial L(b_i, f(a_i))}{\partial f(a_i)}\right]_{f=f_{p-1}^t}$
    fit a CART to target $r_{mi}, j = 1, 2, \ldots .J$
    for $q = 1, 2, 3 \ldots q_p$ **do**
      $\gamma_{mj} = \arg\min_{\gamma} \sum_{x_i \in R_{Pq}} L(y_i, f_{P-1}(a_i) + \gamma), f_{P-1} = f_{P-1}^t$
      $updatef_p(a) = f_{p-1}(a) + \sum_{q=1}^{q_p} \gamma_{pq} I\left(x \in R_{pq}\right), f_p = f_p^t$
    $output : f^t(a) = f_p(a)$
  $updateh_t(a) = h_{t-1}(a) + f^t(a)$
output: $H(a) = \sum_{t=1}^{T} h_t(a)$

check whether to run the periodic tasks first or to run some unnecessarily starving tasks with closer deadline hence any other non-priority periodic task does not miss its deadline. Starving tasks are such tasks that are waiting for their execution, and are not assigned resources due to exception or system priorities. CM is calculated using task data arriving at the system in addition to historical log data. The GB-DT ensemble prediction model predicts the task execution status (Fig.3) and machine utilization. To predict high priority task's safe execution, slack of each task is calculated afterwards the execution time status is analyzed (complete time vs left time) while any overlaps are removed to predict the safe execution of tasks. To check the feasibility of running a low priority task under given constraints the model learns from the history data that whether executing it might make another high priory task prone to be collapse/fail or not. If the model prediction yields the output 1 that means safe execution is possible else 0 is returned. If the prediction outcome returns 1 the value of CM is set to be 0 indicates that CPU has enough resources to be allocated to the starving, tasks else $CM = 1$ means priority periodic task is executed.

C. PREEMPTION THRESHOLD (PT)

To tackle with the overloaded scenarios where only high priority tasks arrive at the processor the proposed scheduling policy does not allow a low-priority task to preempt a CED task through the use of preemption bit and preemption threshold. In case of overloaded situation if the periodic task comes up with a preemption bit it can interrupt the CED tasks. The preemption threshold is set to confine the maximum starvation-time of NPP tasks. $PB = 1$ means that in overloaded scenarios the periodic tasks can preempt the running task and can be executed right away.

IV. TASK MODELING AND SIMULATION SETUP (Case SCENARIO; CANDY-BOX FACTORY)

This section presents a detailed description of task modeling and simulation setup for the developed system. A smart factory comprises heterogeneous smart machine networks owning unique functionalities and working. The machines work together to manufacture various product. Each machine is given a task to complete to achieve the end goal. The manufacturing tasks are assigned to each machine for execution using the scheduling module. The actuators in smart factory are categorized as manufacturing machines and the environment control actuators. Environment control is achieved by turning on/off and minimizing/maximizing the desired environmental control actuators that include heater, chiller, humidifier and dehumidifier. In Table 2 the details of triggered control task as a result of environment sensing data for candy-box factory are presented. The ambient environmental factors affect the machine performance thus the production process become prone to performance degradation, therefore, the execution of control tasks at respective machine alongside controlling the ambient environment is necessary for the smooth production process.

A. TASKS DATASET

This subsection provides the details of the task generation and input task parameters. Every task comprises of several processes for execution on a device. The data acquired from sensors at preset intervals (10 seconds) to conduct series of experiments. The prediction results are fed as an input to the scheduling model along with system tasks and constraints, sensing tasks, system exceptions and user requirements. The process begins with data collection and dataset generation through simulation iterations to generate tasks and schedule them. The data is simulated based on inputs provided by the user and system threshold. The dataset provides historical task data for training the machine learning model for learning assisted scheduling. The initial task parameters task id, execution and deadline time. Based on the initial parameters the system computes the rest of task parameters. Thus, historical task data include information related to arrival time, deadline time, execution time, finish time and time-budget allocated to that task. The machine attributes involve allocated machine, load at the machine and machine capacity.

Furthermore, simulated tasks data is presented in Fig.4. Firstly, the system takes the number of tasks to be generated as input from the user. In the next step user enters task sensing interval with a range of 5 to 60 seconds. The generated tasks interval is set according to user choice. Each task holds some

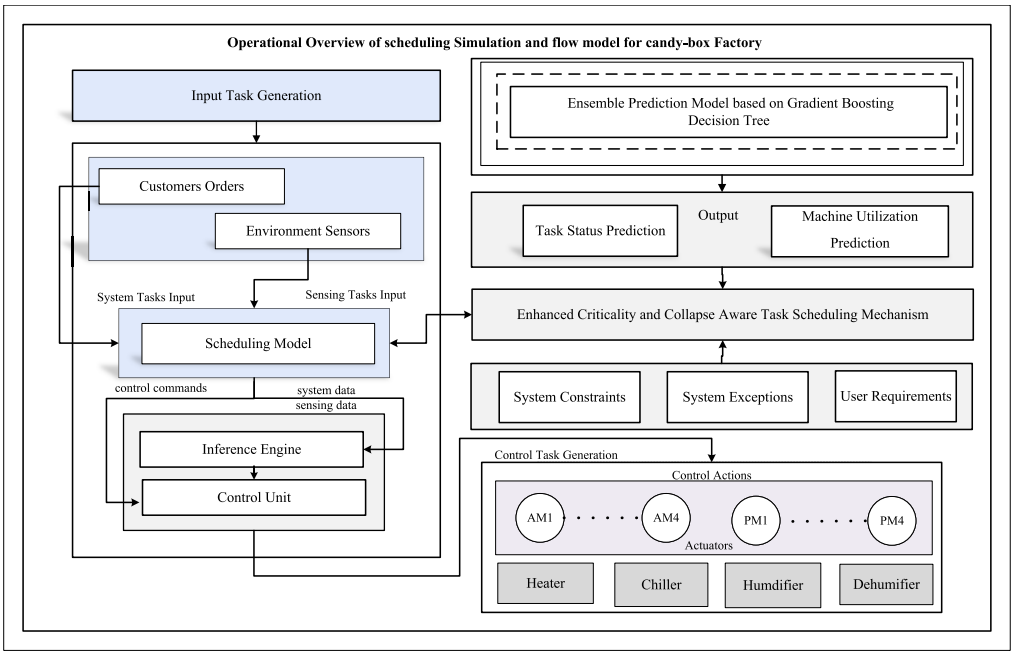

FIGURE 3. Modeling scenario and operational overview for candy-box manufacturing.

TABLE 2. Control task generation for ambient environment control of candy-box factory through actuator control commands.

Type of sensor	Sensor Name	Environment constraints	Sensing value	Actuator Controls	Output status
Environment Ambient sensors	Temperature sensor	19 - 25 °C	Temp >25°C	Turn on Chiller & Increase Chilling Level	1
				Turn off Heater	0
			Temp <19°C	Turn on heater & Increase heating Level	1
				Turn off chiller	0
	Humidity sensor	35% - 50%	Humidity <35%	Turn on humidifier to increase humidity Level	1
				Turn off dehumidifier	0
			Humidity >50%	Turn on dehumidifier to decrease humidity Level	1
				Turn off humidifier	0

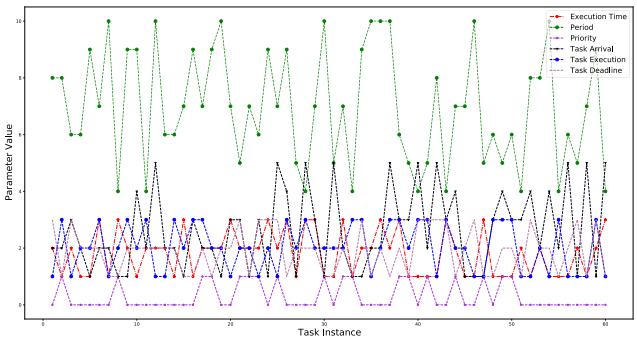

FIGURE 4. Simulated task data for candy-box manufacturing case scenario.

initial parameters used by the system for computing task parameters. A unique task id is assigned to each task with an arrival time and deadline time information associated. The task deadline time is set to be greater than execution time of that task. Machine ID is initialized from zero then later set according to the ID of scheduled machine. Start and finish time corresponds to the scheduled task start and finish time at a particular machine. The process begins with task set generation using the initial parameters afterwards the systems call the task attribute generation function for assigning start time, finish time and time-budget to generated task set. Based on the task parameters the tasks become ready to be schedule and executed according to the proposed scheduling policy. Alongside the scheduling of task, the scheduler maintains a history-log containing derived parameters such as completion status of the task, load at each machine in terms of task processing capacity according to current load.

The simulated data for temperature and humidity for training is presented in Fig. 5. The values of temperature and humidity showed a varied distribution with few exceptions. Several random fluctuations are knowingly inserted to enable the rule engine detect and control outlier samples.

The detailed implementation of the developed system is presented in subsections. The scheduling framework through real time control scheme achieves independent machine and environment control. The predictive analytic assist the scheduling model in triggering flexible responses to periodic and non-periodic happening of smart manufacturing system. Tasks/jobs reach scheduling module along with resource requirements for orderly task allocation to machines by the Scheduler. To evaluate the proposed approach, we devised a use-case scenario of a smart factory involving candy-box packaging and assembling. The motivation behind the consideration of the present study has been taken from a case study [56]. It is assumed that N types of candies are being

TABLE 3. List of task processing time and task type.

Task Description	Processing Time Requirement (ms)	Task Type
Humidity sensing	10	NPP
Temperature sensing	10	NPP
Occupancy sensing	10	NPP
Order placing (system task)	300	CED
Rule execution (system task)	300	PP/NPP
Chilling (actuator control)	520	CED
Heating (actuator control)	520	CED
Humidification (actuator control)	520	NCED/CED
De-humidification (actuator control)	520	NCED/CED
Control machines (AM1-AM4)	520	NCED/CED
Control machines (PM1-PM4)	520	NCED/CED

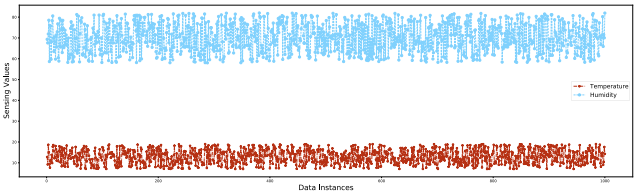

FIGURE 5. Simulated environment sensing data for candy-box manufacturing case scenario.

manufactured in smart candy manufacturing factory to make customized orders as per user orders. Yet the devised task scheduling mechanism does not take account of the manufacturing process of candies for the use-case scenarios. The scenario considers a case where there are N types of candies, and the task is to get customized user orders to assemble and pack the boxes accordingly for delivery.

Furthermore, our proposed case study encompasses two types of sensors (ambient and on-machine): four actuators and eight machines. The details of sensors and actuators are presented in Table 3. The ambient sensors are used to sense the environment parameters every 10ms, specifically humidity, temperature, and occupancy (through on-machine sensors) having task type normal priority periodic task. The actuators control task is sent to the heater, chiller, humidifier, dehumidifier, and manufacturing machines for execution. The processing time requirement of a system task is set as 300 ms with task type categorized as critical event-driven for order placement task. On the other hand, rule execution task priority is set as periodic-priority. All control tasks are given 520 ms to complete their execution. The control task's priorities to be executed at manufacturing machines (AM, PM) are assigned based on user order time and respective deadline. The priority can either be non-critical event-driven or critical event driven.

B. EVENT DRIVEN TASK SIMULATION FLOW

This subsection presents the event-driven task simulation flow and task generation for the candy box factory case scenario. The process is initiated by the user-customized candy box order placement hence the task flow begins with an event-driven task as shown in event driven task simulation flow Fig. 6. The system has now an order placement task after the user places an order for a candy box. Once the system task gets into the pipeline, a corresponding

Table 1

System Task	Task Type	Task Modeling	Order Details
T1 Customized Candy box order	CED	$\{T_{id}, T_{at}, T_{et}, T_{dt}, T_{ft}, T_{lft}, T_{pr}, C_{bit}\}$	Candy type & Candy quantity

Table 2

System Task	Task Type	Number of jobs	Job 1	Job 2
T2 generate response to Candy box order	NCED/CED	2	Assemble Candy-box	Packaging Candy-box

Table 3

Job List	Job Model
J1 Assembling Candy box order	$\{J_1; J_{st}, J_{et}, J_{dt}, J_{ft}, J_{lft}\}$
J2 Packaging Candy box order	$\{J_2; J_{st}, J_{et}, J_{dt}, J_{ft}, J_{lft}\}$

Table 4

Job List	Machine	Job Input + Order Input	Output Status
J1 Assembling Candy box order	AM1..AM4	$\{J_1; J_{st}, J_{et}, J_{dt}, J_{ft}, J_{lft}, Order_{no}, AM_{id}, Candy_{type}, Candy_{quantity}\}$	Candy type & Candy quantity
J2 Packaging Customized Candy box order	PM1..PM4	$\{J_2; J_{st}, J_{et}, J_{dt}, J_{ft}, J_{lft}, Order_{no}, PM_{id}, Packaging_{style}\}$	Candy type & Candy quantity

FIGURE 6. CED task simulation flow for candy-box manufacturing case scenario.

response task is generated by the system, and the generated response is also a system task having two jobs assembling and packing represented by (J1 and J2) in Fig. 6. J1 can be executed on one of the four assembling machines while all four packing machines are available for J2 execution. The selection of machines for the execution of J1(assembling) and J2(Packaging) is made based on the machine's current status in terms of load.

C. PERIODIC TASK SIMULATION FLOW

In this subsection the periodic task simulation is described for generation of periodic tasks for candy box factory. Periodic tasks trigger the sensing tasks. Temperature, humidity and occupancy is sensed every 10ms; the sensors are deployed in candy-box factory for sensing environmental parameters.

The periodic tasks initiate a corresponding inference rule execution task in response to sensing tasks. Inference engine maps the values to a given threshold and generates control commands. For instance in response to occupancy sensing observations the next task might be generation of machine idle alert. The aforementioned phenomena is illustrated in Fig. 7. Likewise the temperature and humidity sensing values might trigger the desired environmental control task in form of inference rule execution of heating, chilling, humidification and de-humidification control. Initially, the sensing tasks are generated using initial parameters afterwards the task generation function is called to create task parameters. Finally, based on the task parameters, the tasks are ready to be executed at the scheduler according to the CCAS policy.

Table 1

System Task	Task Type	Task Modeling	Task Description
T1 Temperature sensing T2 Humidity sensing T3 occupancy sensing	PP	$\{T_{id}, T_{at}, T_{et}, T_{pr}, P_{bit}\}$ $\{T_{id}, T_{at}, T_{et}, T_{pr}, P_{bit}\}$ $\{T_{id}, T_{at}, T_{et}, T_{pr}, P_{bit}\}$	Sensing

Table 2

System Task	Task Type	Task Modeling	Task Description
T1 Inference Rule Execution	PP	$\{T_{id}, T_{at}, T_{pr}, P, P_{bit}\}$	Sensing, Threshold
T2 Machine Idle Alert	ED	$\{T_{id}, T_{at}, T_{pr}, C_{bit}\}$	Machine ID & Status

Table 3

Control Task	Task types	Task Modeling	Description
T1 Chiller Control	NCED/CED	$\{T_{id}, T_{at}, T_{pr}, C_{bit}\}$	Control (ON/OFF) High /Low
T2 Heater Control	NCED/CED	$\{T_{id}, T_{at}, T_{pr}, C_{bit}\}$	Control (ON/OFF) High /Low
T3 Humidification Control	NCED/CED	$\{T_{id}, T_{at}, T_{pr}, C_{bit}\}$	Control (ON/OFF) High /Low
T4 De-humidification Control	NCED/CED	$\{T_{id}, T_{at}, T_{pr}, C_{bit}\}$	Control (ON/OFF) High /Low

FIGURE 7. PP task simulation flow for candy-box manufacturing case scenario.

TABLE 4. Implementation environment.

#	Component of system	Description
1	Hardware	Raspberry Pi 4 (Model B)
2	Operating System	Ubuntu 20.04, Windows 10
3	Memory	8 GB,12 GB
4	Server	Flask Server
5	Resources	Temperature sensor, connecting wires, breadboard and Misc.
6	Libraries	Sklearn SciPy, Numpy, Pandas, Keras
7	IDE	PyCharm Professional
8	CPU	Intel ®Core ™ i5-4500 CPU at 3.40 GHz
9	Database	MySQL

V. EXPERIMENTAL RESULTS AND ANALYSIS

This section presents the experimental results and performance analysis of the developed predictive task scheduling mechanism based on proposed CCAS scheme for ultra-efficient resource load balance in smart factory. The proposed scheduling framework has been developed using python as it is widely employed for implementation of various applications specifically web and desktop. Further details of the development environment are described in the Table 4.

Fig. 8 presents the prediction results of the ensemble prediction model for task status prediction on train and test data. The ensemble prediction model is developed based on Gradient boosting decision tree algorithm. The ensemble model harness the generalization abilities of GBDT; built sequentially for reducing the biases of combined estimator.

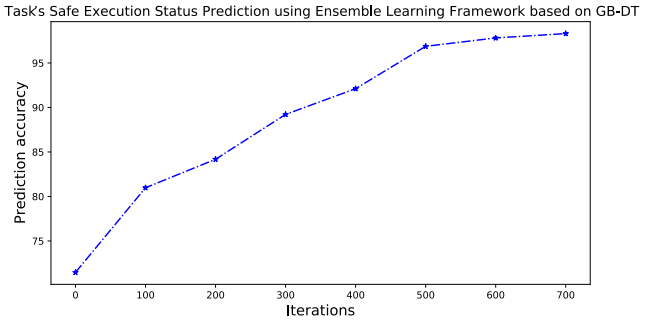

FIGURE 8. Task execution status prediction using GBDT.

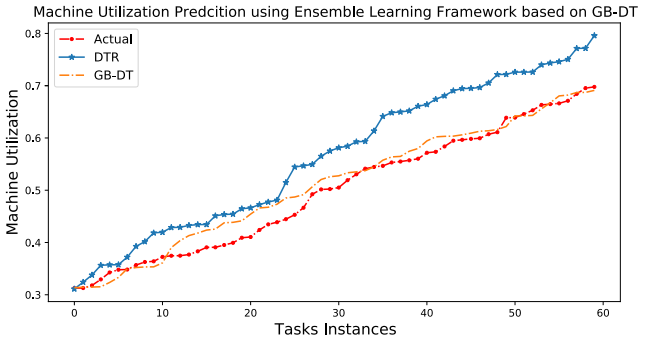

FIGURE 9. Machine utilization prediction using GBDT and baseline model.

The base learners are trained sequentially lastly the final model provides prediction outcomes by aggregating results of the each-step. The key parameters for ensemble GBDT model corresponds to learning rate and number of estimators. The optimal number of trees per round is 50, while the optimal max-tree depth is found at 6. Model hyper-parameters are tuned using Bayesian Optimization Hyper-band (BOHB) to achieve accurate prediction results.

The ensemble model predicts task status and machine utilization based on learning task parameters as discussed earlier in. Algorithm 2. To evaluate the effectiveness of proposed ensemble model in terms of performance; a comparative analysis has been performed with the baseline Decision Tree algorithm. Fig. 9 shows the actual and predicted machine utilization by proposed ensemble model as well as stand-alone Decision Tree model. The prediction results of the proposed ensemble model show significant improvements in terms of accuracy compared to the standalone model. The accuracy achieved by the proposed ensemble model is 98.37% in 700 iterations for task status prediction. While the maximum accuracy achieved by the ensemble model in case of machine utilization is 97.91% in 500 iterations. In contrast the stand alone decision tree model attained a prediction accuracy of 96.48% and 94.45% for task status and machine utilization respectively. The performance achieved by the proposed ensemble model is attributed to strategy adopted for optimal hyper-parameter optimization, faster convergence and higher efficiency of GB-DT model. The results demonstrate that the ensemble model performed well on average.

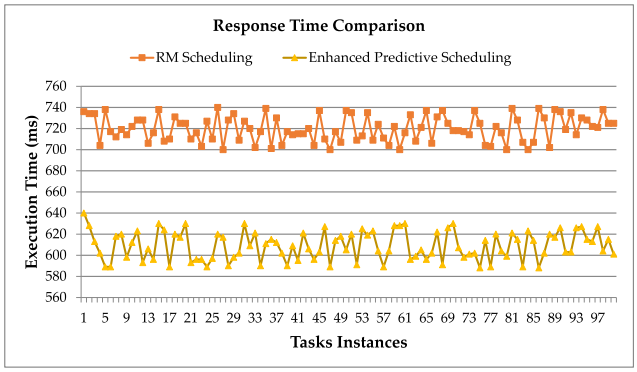

FIGURE 10. Tasks response time analysis and comparison.

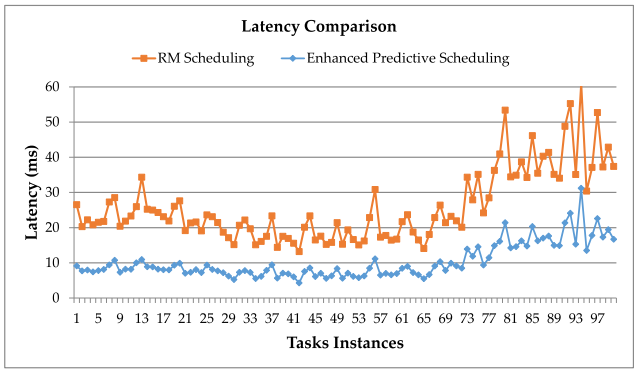

FIGURE 11. Tasks latency analysis and comparison.

In Fig. 10 a comparative analysis has been performed with baseline RM-scheduling approach to demonstrate the effectiveness of learning-assisted scheduling algorithm.

The performance is analyzed based on response time, latency, and machine utilization in accordance to the goal of the system that is maximizing machine utilization and optimal resource utilization. The response time comparison in Fig. 10, between classic RM scheduling and proposed predictive scheduling is presented. The plot shows execution time on the y-axis while task instances are plotted on the x-axis. It is evident from the comparative analysis that the proposed CCAS mechanism improved the allocation of task to machines which resulted into enhanced machine utilization and improved response time. The proposed scheduling scheme achieved superior performance in terms of average response time in contrast to the baseline scheduling algorithm due to intelligent decision-making capabilities. The average response time analysis of the classic RM algorithm is 720 ms and the average response time analysis for proposed predictive scheduling is 609 ms. Hence, our proposed enhanced scheduling mechanism improved machine utilization by efficiently allocating tasks to machines compared to the classic RM scheduling algorithm.

For comparative analysis Fig. 11 presents the scheduling comparison between classic RM scheduling and the enhanced predictive scheduling. The graph shows the latency on the y-axis and test instances on the x-axis. The latency analysis graph shows that the proposed predictive scheduling has lower latency than the classic RM scheduling algorithm.

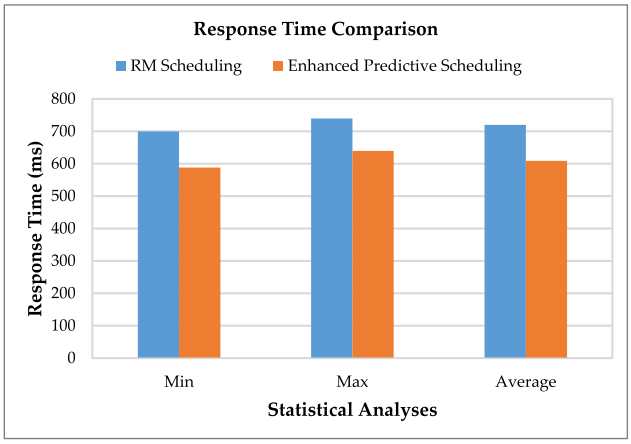

FIGURE 12. Response time comparison of the predictive scheduling and RM scheduling.

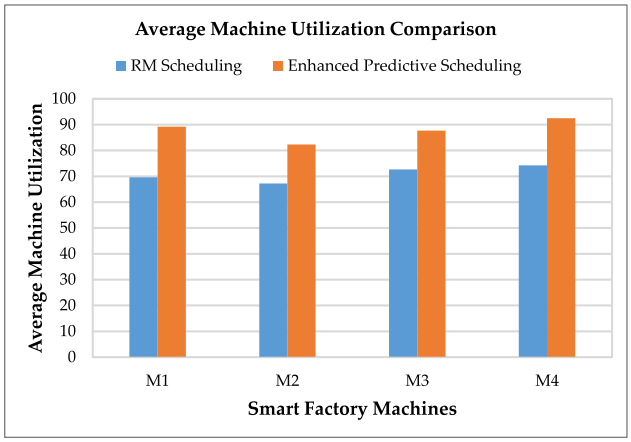

FIGURE 13. Average machine utilization comparison of predictive scheduling and RM scheduling.

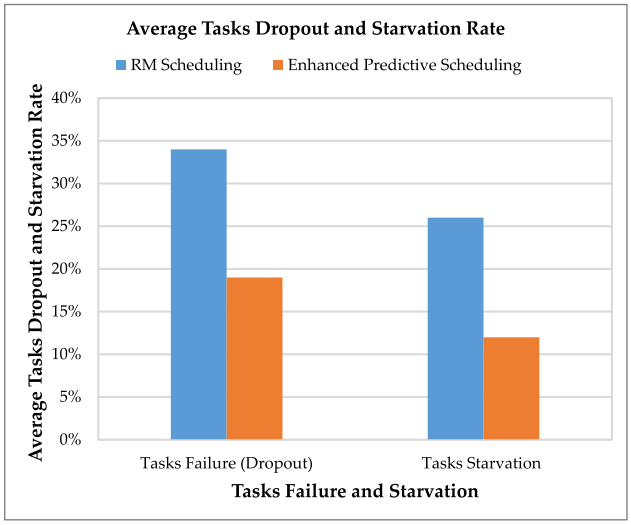

FIGURE 14. Tasks dropout and starvation analysis.

Hence our proposed predictive scheduling reduced the waiting time of tasks in the execution queue, which is only possible due to the criticality and collapse task-aware scheduling strategy of the proposed predictive scheduling mechanism. In Fig. 12, we presented a response time comparison between

FIGURE 15. Tasks simulation sequence while executing at machines for candy-box factory.

classic RM scheduling and proposed predictive scheduling based on enhanced predictive RM scheduling with criticality and collapse aware based priority assignment to tasks. The graph shows the response time in milliseconds on the y-axis and the minimum, maximum, and average response time values. The comparative analysis of the response time shows that the enhanced predictive scheduling has improved results than the classic RM algorithm due to its ability to maximize machine utilization. The proposed predictive scheduling scheme's minimum, maximum and average response time is 588 ms, 640 ms, and 609 ms, respectively. The maximum response time of classic RM scheduling is 740 ms because it does not guarantee the optimal solution in different task periods and deadlines. The statistical analysis demonstrates that our proposed scheduling mechanism improved the overall task execution process in a smart factory.

Furthermore, Fig. 13 shows the average machine utilization-based comparative analysis between classic RM scheduling and enhanced predictive scheduling. Average machine utilization is plotted on the y-axis, while the x-axis depicts the four assembly machines. It is evident from the graphical representation values that the average machine utilization output by proposed predictive scheduling is higher than the classic method due to the integration of a learning module that aids the smart factory task scheduling through accurate predictions of task status and machine utilization.

The predictions facilitate the scheduler in intelligent decision-making and thus improves the load balance and overall performance of the smart factory.

In addition, Fig. 14 shows task starvation analysis and dropout rate analysis. The graph shows the average task starvation and dropout rate percentage on the x-axis. A comparative study between the classic RM algorithm and proposed predictive scheduling shows that the number of tasks starvation rate and missed task rate is lower in the case of enhanced predictive CCAS.

For scheduling simulation, analysis and visualization we developed a web powered visualization tool kit as front end interface. Fig. 15 shows the output of the developed scheduling scheme depicting the execution of tasks at machines.

VI. CONCLUSION

The average task dropout or failure rate achieved by improved predictive scheduling is 19%, while the average task starvation is 12%. In contrast, the average task drop out of the conventional RM scheduling algorithm is 34%, and the task starvation rate is 26%. The comparative analysis demonstrates that the proposed scheduling mechanism improved the overall performance of the smart machines compared to the conventional RM scheduling algorithm. The reason for high task starvation and failure is because, in classic RM scheduling, the machine utilization is lower than expected. In contrast, the proposed enhanced predictive scheduling due to its improved scheduling policy, integration of learning and control module leads to utilization of machines to its fullest.

In this study we proposed a learning-assisted scheduling scheme for scheduling task in constrained smart candy-box manufacturing environment. The proposed work presents an enhanced task criticality-aware scheduling algorithm capable of handling various combinations of task types and priorities task in constrained manufacturing environment. The system is developed using three modules; namely task scheduling mechanism based on task criticality and collapse aware strategy, An ensemble GBDT prediction model for providing predictive analytics and an optimal environmental control based on inference rule engine. Furthermore a webpower visualization toolkit is also developed for visualizing the output of the task while executing at machines. The developed system makes the best effort in minimizing the task starvation rate in overloaded scenarios by fair allocation of machine resource to task with the help of two intelligent measure namely CFM and CM. Moreover the proposed system involve elements such as reserve, preemption bit and preemption threshold for task scheduling under constraints. The main objective of the proposed mechanism is to maximize the dynamism, efficiency of resources utilization, flexibility and re-configurability of a constrained production processes in real time. The ensemble learning framework assists the task scheduling mechanism in achieving efficient resource load balance in a smart factory by predicting task scheduler's next move. An enhanced CCAS method is employed that focus productivity maximization. The developed framework is evaluated on task modeling and simulation scenario for candy-box factory. For comparative analysis the performance of the proposed system is compared with baseline models. From the experimental results it is concluded that the proposed CCAS scheme achieved improved task dropout and starvation rate by 15% and 14% respectively leading to improved resource utilization of resources. Future work involves integration of optimization module to further enhance the performance of the proposed system to ensure error-free tasks execution. Furthermore in future work we aim to establish a comparative analysis with several more

competitive scheduling schemes to verify the applicability of the future proposed scheduling mechanism in presence of other schedulers.

REFERENCES

[1] K. Nagorny, P. Lima-Monteiro, J. Barata, and A. W. Colombo, "Big data analysis in smart manufacturing: A review," *Int. J. Commun., Netw. Syst. Sci.*, vol. 10, no. 3, pp. 31–58, 2017.

[2] Q.-L. Guo and M. Zhang, "An agent-oriented approach to resolve scheduling optimization in intelligent manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 26, no. 1, pp. 39–45, Feb. 2010.

[3] E. Hopkins and A. Siekelova, "Internet of Things sensing networks, smart manufacturing big data, and digitized mass production in sustainable industry 4.0," *Econ., Manage. Financial Markets*, vol. 16, no. 4, pp. 2–41, 2021.

[4] J. Wan, M. Yi, D. Li, C. Zhang, S. Wang, and K. Zhou, "Mobile services for customization manufacturing systems: An example of industry 4.0," *IEEE Access*, vol. 4, pp. 8977–8986, 2016.

[5] L. Zhou, L. Zhang, and B. K. P. Horn, "Deep reinforcement learning-based dynamic scheduling in smart manufacturing," *Proc. CIRP*, vol. 93, pp. 383–388, Jan. 2020.

[6] C. Chiu and Y. Yih, "A learning-based methodology for dynamic scheduling in distributed manufacturing systems," *Int. J. Prod. Res.*, vol. 33, no. 11, pp. 3217–3232, Nov. 1995.

[7] A. Rajabinasab and S. Mansour, "Dynamic flexible job shop scheduling with alternative process plans: An agent-based approach," *Int. J. Adv. Manuf. Technol.*, vol. 54, nos. 9–12, pp. 1091–1107, 2001.

[8] P. Fattahi and A. Fallahi, "Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability," *Cirp J. Manuf. Sci. Technol.*, vol. 2, no. 2, pp. 114–123, 2010.

[9] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Inf. Sci.*, vol. 298, no. 219, pp. 198–224, Mar. 2015.

[10] M. A. Adibi, M. Zandieh, and M. Amiri, "Multi-objective scheduling of dynamic job shop using variable neighborhood search," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 282–287, 2010.

[11] C. Pelau, D.-C. Dabija, and I. Ene, "What makes an AI device human-like? The role of interaction quality, empathy and perceived psychological anthropomorphic characteristics in the acceptance of artificial intelligence in the service industry," *Comput. Hum. Behav.*, vol. 122, Sep. 2021, Art. no. 106855.

[12] Y. Li, S. Carabelli, E. Fadda, D. Manerba, R. Tadei, and O. Terzo, "Machine learning and optimization for production rescheduling in industry 4.0," *Int. J. Adv. Manuf. Technol.*, vol. 110, nos. 9–10, pp. 2445–2463, Oct. 2020.

[13] A.-N. Khan, N. Iqbal, A. Rizwan, R. Ahmad, and D.-H. Kim, "An ensemble energy consumption forecasting model based on spatial-temporal clustering analysis in residential buildings," *Energies*, vol. 14, no. 11, p. 3020, May 2021.

[14] S. K. Kiangala and Z. Wang, "An effective adaptive customization framework for small manufacturing plants using extreme gradient boosting-XGBoost and random forest ensemble learning algorithms in an industry 4.0 environment," *Mach. Learn. With Appl.*, vol. 4, Jun. 2021, Art. no. 100024.

[15] A. Petrakova, M. Affenzeller, and G. Merkurjeva, "Heterogeneous versus homogeneous machine learning ensembles," *Inf. Technol. Manage. Sci.*, vol. 18, no. 1, pp. 135–140, Jan. 2015.

[16] Y.-H. Hung, "Improved ensemble-learning algorithm for predictive maintenance in the manufacturing process," *Appl. Sci.*, vol. 11, no. 15, p. 6832, Jul. 2021.

[17] A. N. Khan, N. Iqbal, R. Ahmad, and D.-H. Kim, "Ensemble prediction approach based on learning to statistical model for efficient building energy consumption management," *Symmetry*, vol. 13, no. 3, p. 405, Mar. 2021.

[18] G. H. Popescu, S. Petreanu, B. Alexandru, and H. Corpodean, "Internet of Things-based real-time production logistics, cyber-physical process monitoring systems, and industrial artificial intelligence in sustainable smart manufacturing," *J. Self-Governance Manage. Econ.*, vol. 9, no. 2, pp. 52–62, 2021.

[19] M. K. Sohrabi and H. Azgomi, "A survey on the combined use of optimization methods and game theory," *Arch. Comput. Methods Eng.*, vol. 27, no. 1, pp. 59–80, Jan. 2020.

[20] R. X. Gao, L. Wang, M. Helu, and R. Teti, "Big data analytics for smart factories of the future," *CIRP Ann.*, vol. 69, no. 2, pp. 668–692, 2020.

[21] S. Malik, K. Lee, and D. Kim, "Optimal control based on scheduling for comfortable smart home environment," *IEEE Access*, vol. 8, pp. 218245–218256, 2020.

[22] D. Ivanov, A. Dolgui, B. Sokolov, F. Werner, and M. Ivanova, "A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0," *Int. J. Prod. Res.*, vol. 54, no. 2, pp. 386–402, Jan. 2016.

[23] S. Ahmad and D. Kim, "Design and implementation of thermal comfort system based on tasks allocation mechanism in smart homes," *Sustainability*, vol. 11, no. 20, p. 5849, 2019.

[24] A. Novak, D. Bennett, and T. Kliestik, "Product decision-making information systems, real-time sensor networks, and artificial intelligence-driven big data analytics in sustainable industry 4.0," *Econ., Manage. Financial Markets*, vol. 16, no. 2, pp. 62–72, 2021.

[25] M. Nouiri, A. Bekrar, and A. Jemai, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 603–615, Mar. 2018.

[26] S. Malik, S. Ahmad, I. Ullah, D. H. Park, and D. Kim, "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*, vol. 11, no. 8, p. 2192, Apr. 2019.

[27] N. Iqbal, Imran, S. Ahmad, R. Ahmad, and D.-H. Kim, "A scheduling mechanism based on optimization using IoT-tasks orchestration for efficient patient health monitoring," *Sensors*, vol. 21, no. 16, p. 5430, Aug. 2021.

[28] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Gener. Comput. Syst.*, vol. 110, pp. 1098–1115, Sep. 2020.

[29] H. R. Lewis, "Review: Michael R. Garey, David S. Johnson, Computers and intractability: A guide to the theory of *NP*-completeness," *J. Symbolic Log.*, vol. 48, no. 2, pp. 498–500, 1983.

[30] Z. Peng, D. Cui, J. Zuo, Q. Li, B. Xu, and W. Lin, "Random task scheduling scheme based on reinforcement learning in cloud computing," *Cluster Comput.*, vol. 18, pp. 1595–1607, Sep. 2015.

[31] H. Zhang and G. He, "An adaptive task scheduling system based on real-time clustering and NetFlow prediction," in *Proc. IEEE 5th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2019, pp. 77–80.

[32] D. Cui, Z. Peng, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or IaaS cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1030–1039, Nov. 2020, 10.1109/TCC.2017.2773078.

[33] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, Atlanta, GA, USA, Nov. 2016, pp. 50–56.

[34] S. Swarup, E. M. Shakshuki, and A. Yasar, "Task scheduling in cloud using deep reinforcement learning," *Proc. Comput. Sci.*, vol. 184, pp. 42–51, Jan. 2021.

[35] J. Townsend, "Interconnected sensor networks and machine learning-based analytics in data-driven smart sustainable cities," *Geopolitics, Hist., Int. Relations*, vol. 13, no. 1, pp. 31–41, 2021.

[36] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6069–6078, Sep. 2020.

[37] M. Kovacova and G. Lăzăroiu, "Sustainable organizational performance, cyber-physical production networks, and deep learning-assisted smart process planning in industry 4.0-based manufacturing systems," *Econ., Manage. Financial Markets*, vol. 16, no. 3, pp. 41–54, 2021.

[38] E. Badidi, "QoS-aware placement of tasks on a fog cluster in an edge computing environment," *J. Ubiquitous Syst. Pervasive Netw.*, vol. 13, no. 1, pp. 11–19, Oct. 2020.

[39] T. Dong, F. Xue, C. Xiao, and J. Li, "Task scheduling based on deep reinforcement learning in a cloud manufacturing environment," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 11, p. e5654, 2020.

[40] H. Chen, G. Liu, S. Yin, X. Liu, and D. Qiu, "ERECT: Energy-efficient reactive scheduling for real-time tasks in heterogeneous virtualized clouds," *J. Comput. Sci.*, vol. 28, pp. 416–425, Sep. 2018.

[41] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[42] D. E. Akyol and G. M. Bayhan, "A review on evolution of production scheduling with neural networks," *Comput. Ind. Eng.*, vol. 53, no. 1, pp. 95–122, Aug. 2007.

[43] B. Sokolov and D. Ivanov, "Integrated scheduling of material flows and information services in industry 4.0 supply networks," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1533–1538, 2015.
[44] S. Nakasuka and T. Yoshida, "New framework for dynamic scheduling of production systems," in *Proc. Int. Workshop Ind. Appl. Mach. Intell. Vis.*, 1989, pp. 253–258.
[45] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.
[46] J. Wan, X. Li, H.-N. Dai, A. Kusiak, M. Martínez-García, and D. Li, "Artificial-intelligence-driven customized manufacturing factory: Key technologies, applications, and challenges," *Proc. IEEE*, vol. 109, no. 4, pp. 377–398, Apr. 2021.
[47] M. Ghahramani, Y. Qiao, M. Zhou, A. O. Hagan, and J. Sweeney, "AI-based modeling and data-driven evaluation for smart manufacturing processes," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 4, pp. 1026–1037, Jul. 2020.
[48] L. Wang, "From intelligence science to intelligent manufacturing," *Engineering*, vol. 5, no. 4, pp. 615–618, Aug. 2019.
[49] J. Wang, J. Yan, C. Li, R. X. Gao, and R. Zhao, "Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction," *Comput. Ind.*, vol. 111, pp. 1–14, Oct. 2019.
[50] G. Jules and M. Saadat, "Agent cooperation mechanism for decentralized manufacturing scheduling," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 12, pp. 3351–3362, Dec. 2017.
[51] D. A. Rossit, F. Tohmé, and M. Frutos, "Industry 4.0: Smart scheduling," *Int. J. Prod. Res.*, vol. 57, no. 12, pp. 3802–3813, Jun. 2019.
[52] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *J. Manuf. Syst.*, vol. 48, pp. 157–169, Jul. 2018.
[53] V. Vlad, "Holonic-based task scheduling in smart manufacturing systems," in *Proc. SMARTGREENS*, 2019, pp. 242–245.
[54] P. Priore, B. Ponte, J. Puente, and A. Gómez, "Learning-based scheduling of flexible manufacturing systems using ensemble methods," *Comput. Ind. Eng.*, vol. 126, pp. 282–291, Dec. 2018.
[55] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.
[56] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

ANAM NAWAZ KHAN received the B.S. and M.S. degrees in computer science from COMSATS University Islamabad, Pakistan, in 2016 and 2019, respectively. In the meantime, she joined various reputable academic institutions and later decided to expand her knowledge by continuing her academic career. In 2020, she moved to the Republic of Korea and started working as a Ph.D. Research Fellow with the Department of Computer Engineering, Jeju National University. Her research interests include development and optimization of artificial intelligence and machine learning based applications in smart spaces. She is currently working on integrating the AI enabled IoT Platforms and edge computing to provide autonomous intelligent services for smart environments.

NAEEM IQBAL received the B.S. and M.S. degrees in computer science from COMSATS University Islamabad, Attock Campus, Punjab, Pakistan, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Jeju National University, Republic of Korea. He has professional experience in the software development industry and in academic as well. His research work mainly focused on AI-based intelligent systems, data science, big data analytics, machine learning, deep learning, analysis of optimization algorithms, the IoT, and blockchain-based secured applications. He has published more than 20 papers in peer-reviewed international journals and conferences. He is serving as a professional reviewer for various well-reputed journals and conferences.

ATIF RIZWAN received the M.S. and Master of Computer Science degrees in computer science from COMSATS University Islamabad, Attock Campus, Punjab, Pakistan, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Jeju National University, Republic of Korea. He has good industry experience in software development and testing. His research interests include machine learning, data and web mining, analysis and optimization of core algorithms, and the IoT-based applications.

SEHRISH MALIK received the B.S. degree in computer sciences from the FAST-NUCES, Islamabad, the M.S. degree from Sangmyung University at Cheonan Campus, supervised by Prof. J.-H. Lee, and the Ph.D. degree from the Mobile Computing Laboratory (MCL), Jeju National University, under the supervision of Prof. D.-H. Kim. She successfully defended her Ph.D. Thesis, in December 2019. She recently joined the Data Driven Software Engineering Department, Simula Research Laboratory, Oslo, Norway, where she is currently an ERCIM Research Fellow. Her research interest includes machine learning applications in smart environments. She was a member of the Protocol Engineering Laboratory (PEL).

RASHID AHMAD received the B.S. degree from the University of Malakand, Pakistan, the M.S. degree in computer science from the National University of Computer and Emerging Sciences (FAST-NUCES), Islamabad, Pakistan, and the Ph.D. degree in computer engineering from Jeju National University, South Korea, in 2015. He is currently working as a Brainpool Research Fellow with the Bigdata Research Center, Jeju National University. He is on research leave from COMSATS University Islamabad, Attock Campus, where he working as an Assistant Professor. His research work spans around the applied machine learning and the application of prediction and optimization algorithms for building IoT-based edge intelligence solutions. His research interests include machine learning, distributed machine learning, continual learning, and optimization of deep learning models.

DO HYEUN KIM received the B.S. degree in electronics engineering and the M.S. and Ph.D. degrees in information telecommunication from Kyungpook National University, South Korea, in 1988, 1990, and 2000, respectively. He was with the Agency of Defense Development (ADD), from 1990 to 1995. Since 2004, he has been with Jeju National University, South Korea, where he is currently a Professor with the Department of Computer Engineering. From 2008 to 2009, he was a Visiting Researcher with the Queensland University of Technology, Australia. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service, and mobile computing.