Anbar Journal of Engineering Science (AJES) **Vol. 16 , No. 1 (2025) , pp. 21 ~ 35**
P-ISSN: 1997-9428 ; E-ISSN: 2705-7440 DOI: http://doi.org/10.37649/aengs.2025.156372.1120

University of Anbar

Anbar Journal of Engineering Science

journal homepage: https://ajes.uoanbar.edu.iq/

Optimizing Cloud-Edge Integration for Task Scheduling in Smart Manufacturing Lines: A Multi-objective Method

Ahmed Najat Ahmed[a], Mohammed Adam[b], Ari Taha Guron[c], Ali Hasan Hussein[d]
[a]*Department of Information Technology, College of Engineering, University of Lebanese French, Erbil, Iraq*
Email: a.afandy@lfu.edu.krd*; ORCID:* https://orcid.org/0000-0003-0116-8377
[b]*Department of Information Technology, College of Engineering, University of Lebanese French, Erbil, Iraq*
Email: mohammed.adam@lfu.edu.krd*; ORCID:* https://orcid.org/0000-0002-1343-1163
[c]*Department of Computer Science, Faculty of Science and Engineering, University of Bayan, Erbil, Iraq*
Email: ari.taha@bnu.edu.iq*; ORCID:* https://orcid.org/my-orcid?orcid=0009-0000-8160-5834
[d]*Department of Management Information Systems, Technical Administrative Institute, Erbil Polytechnic University, Erbil, Iraq*
Email: ali.hussien@epu.edu.iq*; ORCID:* https://orcid.org/my-orcid?orcid=0000-0001-5220-3318

PAPER INFO

Paper history
Received: 05/1/2025
Revised: 09/2/2025
Accepted: 16/02/2025

Keywords:
Smart Manufacturing
Cloud-Edge Computing
Multi-objective Optimization
Genetic Algorithm
Optimization techniques

ABSTRACT

The convergence of cloud and edge computing in smart manufacturing offers significant potential for improving efficiency in Industry 4.0. However, task scheduling in this context remains a complex, multi-objective challenge. This study introduces a novel Cloud-Edge Smart Manufacturing Architecture (CESMA), leveraging a hybrid approach that integrates NSGA-II and the Improved Monarch Butterfly Optimization (IMBO) algorithms. The combination utilizes NSGA-II's global search and non-dominated solution capabilities with IMBO's fine-tuning and local optimization strengths to enhance task scheduling performance. Where CESMA combines the scalability and analytics power of cloud computing with edge-based real-time decision-making to address the dynamic demands of smart manufacturing. Through extensive simulations and experiments, the feasibility and effectiveness of CESMA are validated, showing improved task scheduling quality, resource utilization, and adaptability to changing conditions. This research establishes a robust platform for managing the complexities of task scheduling in cloud-edge environments, advancing intelligent manufacturing processes, and contributing to the integration of evolutionary algorithms for real-time industrial decision-making.

1. Introduction

In the age of Industry 4.0, smart manufacturing (SM) has undergone a transformative paradigm shift through the synergistic use of, and convergence between, cloud and edge computing technologies [1-4]. Amid this amalgamation lie promises of unprecedented efficiency boosts and novel solutions to industrial challenges. Yet, with such a technological revolution, optimizing the schedule in such dynamic environment is still a challenge, considering multiple objectives [5-7].

By combining cloud and edge computing technologies, smart industries have achieved an unmatched capability to utilize the benefits of real-time data processing, scalability, and distributed decision making. The essence of task scheduling in

* Corresponding author: Ali Hasan Hussein *E-mail@ali.hussien@epu.edu.iq* ; +9647504148774

these systems where the processes are well coordinated and the efficiency is crucial cannot be underestimated [8] [9]. First and foremost, an efficient scheduling directly affects production efficiency, resource utilization, energy consumption, and quality assurance. Conventional approaches to task scheduling are generally static and centrally controlled and problematically struggle with the continuously changing complexities of modern smart manufacturing lines. In resolving this, a multi-objective approach seems the most captivating solution [10]. This approach offers a complete picture by tackling different conflicting objectives simultaneously, thus achieving balance amid competing demands [11].

This paper proposes the Cloud-Edge-based Smart Manufacturing Architecture, CESMA, a paradigm shift in scheduling. CESMA integrates two state-of-the-art optimization algorithms: NSGA-II [12-15] and the Improved Monarch Butterfly Optimization algorithm, IMBO [16-18]. NSGA-II does well in exploring scheduling solutions on a global level, while IMBO contributes expertise in local optimization [19]. They provide CESMA with comprehensive scheduling insights that are both globally non-dominated and locally refined for efficiency. This study epitomizes CESMA's effectiveness through extensive simulations and empirical studies, showing a vast improvement in scheduling quality, resource utilization, and adaptability to dynamic manufacturing conditions. More than that, this work takes part in the broader debate on how cloud-edge computing, combined with evolutionary algorithms, can be used for more effective decision-making within real-time industrial environments to promise versatile solutions amid Industry 4.0 revolutions. Because smart manufacturing lines are remote and heterogeneous, optimizing cloud-edge integration for work scheduling poses special difficulties. Tasks must be distributed effectively among cloud and edge resources in these kinds of settings, juggling several competing goals like reducing latency, maximizing resource use, guaranteeing system scalability, and preserving energy efficiency. These complex requirements are frequently beyond the scope of conventional single-objective or non-hybrid optimization approaches. They frequently overlook the complex interactions between these levels to concentrate on either cloud or edge optimization. This constraint provides compelling evidence for implementing a hybrid strategy that combines several algorithmic capabilities to address the intricacy of cloud-edge work scheduling successfully.

The main contributions of the paper are as follows:

- The article presents CESMA, a novel Smart Manufacturing Architecture that integrates cloud and edge computing.
- CESMA combines two efficient optimization algorithms, NSGA-II and IMBO, to deal with the complex task scheduling problem.
- CESMA optimizes task scheduling by effectively managing the conflicting goals of production efficiency, resource utilization, energy consumption, and quality assurance.

Through extensive simulation, CESMA shows an effective way of improving scheduling quality, adaptability, and resource efficiency by large margins—hence, it is an essential tool in the decision-making of Industry 4.0 environments.

2. Literature Review

The problem is optimizing production [20] scheduling and computation offloading in intelligent workshops with a Cloud-Edge-Terminal architecture. This balance optimization considers production efficiency and computing delay, which can solve the strong coupling relationship between the production jobs and computing tasks. The proposed model, PCCO, aims at dual objectives: minimizing total offloading delay time for computing tasks and the maximum completion time for production jobs. An enhanced multi-objective whale optimization algorithm with improved exploration and diversity-preserving mechanisms is used to achieve these dual objectives. The traditional job scheduling faces the challenges of low information transparency, delayed response, inaccurate scheduling, and suboptimal optimization that hinder productivity and competitiveness in an enterprise [21]. It introduces an innovative approach integrating energy consumption concerns into job shop scheduling to address these. It is designed to reduce completion times, delay, and energy consumption, realizing the importance of energy efficiency in modern manufacturing. Formulate a multi-objective scheduling model [22] as a mixed integer linear programming (MILP) problem and utilize preemptive fuzzy goal programming (FGP) with linguistic terms for a solution, emphasizing importance factors. It also introduces novel reinforcement learning (RL) algorithms, including

SARSA, Q-learning, and Deep-Q-Network (DQN), to optimize resource scheduling in CMfg. [23] Address the critical scheduling and process optimization challenges for blockchain-enabled cloud manufacturing (SPO-BCMfg), recognizing its significance in achieving service-oriented goals. Blockchain-enhanced cloud manufacturing offers improved collaboration and information security, integrating distributed storage and consensus mechanisms, making SPO-BCMfg a complex multi-objective optimization problem. The article establishes a dynamic selection evolutionary algorithm to tackle this challenge, focusing on convergence and diversity, demonstrating its superior performance to other advanced evolutionary algorithms.

The article [24] tackles challenges in handling production exceptions within smart manufacturing, dealing with resource uncertainties, delayed identification and control, and prolonged decision-making due to complex factors. It introduces an edge-cloud collaboration-based self-adaptive approach that leverages IoT and edge computing for resource intelligence, uses fuzzy Bayesian networks for exception diagnosis, and promotes self-adaptive handling through machine collaboration across production levels, demonstrated to improve efficiency in a casting post-processing system case study. Challenges posed by the vast expansion [25] of the Internet of Things (IoT) by developing a multi-cloud task scheduling model with six key goals: minimizing time complexity, cost, internet traffic, energy usage, optimizing resource utilization, and achieving load balancing. The study employs a multi-objective intelligent algorithm based on the sine function to maximize these goals, ultimately enhancing scheduling effectiveness and security in IoT data processing, offering a novel solution to IoT's data management challenges. This article [26] tackles workflow applications' rising significance, driven by computing technology advancements. It addresses the challenge of optimizing complex workflows, accounting for factors like Quality of Service, task dependencies, and user deadlines. It introduces the Multi-objective Artificial Algae (MAA) algorithm for efficient scientific workflow scheduling in a fog-cloud environment, focusing on reducing execution times, energy usage, and costs while maximizing fog resource utilization, addressing a gap in heterogeneous computing systems.

3. Methodology

3.1 Smart Manufacturing Lines:

Optimizing various objectives in the context of bright manufacturing lines is paramount, and the whole manuscript has a novel IIoT cloud-edge-end collaborative computing offload architecture. This developed approach capitalizes on the unique strengths of cloud servers, known for their powerful computing and storage resources, and edge servers, prized for their low communication cost, short response time, and robust network adaptability. By harnessing the best of both worlds, the whole manuscript seamlessly integrates and invokes heterogeneous computing resources, dynamically offloading tasks to the most suitable location based on the distinct requirements of different applications. The concept of the system model is adapted from the study [27].

3.1 1 Architecture Overview:

The whole manuscript architecture consists of three pivotal layers, as illustrated in Figure 1.

A. End-Device Layer: The end-device layer is the lowest layer, and it contains many IIoT devices that are abundant in sensors. These are small, sensor-rich devices but are highly constrained by small battery capacity, limited computing resources, and constrained storage resources. An energy-efficient data transmission mechanism is envisaged to process the data mined from these devices. The task data from these devices are forwarded to the edge or cloud layers through wireless access points (APs) or base stations (BSs) for efficient data flow.

B. Edge Layer: The middle layer, also known as the edge layer, is composed of lightweight edge servers strategically placed at the periphery of a network. Indeed, these edge servers are well suited to offering low-latency computing services, which perfectly fit the real-time requirements of smart manufacturing processes. Not wanting to get overloaded and desiring the best allocation of

computing resources, the edge servers can offload tasks wisely to the cloud servers via high-speed wired links so that there is harmony in the computing resource allocation.

C. Cloud Layer: The cloud layer is filled by cloud servers' rich in computing and storage resources that simultaneously serve users in different geographical regions. This introduces data transmission issues because of the multiple users sharing the same resources and the large distances involved in data transmission using these powerful but remote cloud resources. Thus, overcoming the issues is a part of whole manuscript multi-objective optimization.

3.1.2 Multi-objective Approach:

whole manuscript architecture balances the multi-objective requirements of intelligent manufacturing lines using intelligent task offloading considering computing power, energy cost, and latency. The most appropriate layer, whether it's the agile edge for low-latency processing or the robust cloud for intensive computations, is dynamically allocated to the tasks while ensuring minimum energy consumption and efficient utilization of resources. This dynamic task allocation is guided by the imperatives of production efficiency, resource optimization, energy conservation, and quality assurance—hence setting up an agile, responsive, and multi-objective-driven framework for smart manufacturing.

In the following sections, whole manuscript will go into the finer details of this architecture, showing how it can be transformative for smart manufacturing lines. whole manuscript shall show how it shapes these manufacturing environments into agile ecosystems driven by multiple objectives while harvesting the full potential of both cloud and edge computing. Elaborating on top of the models presented in the study [27], whole manuscript will further detail the proposed CESMA approach.

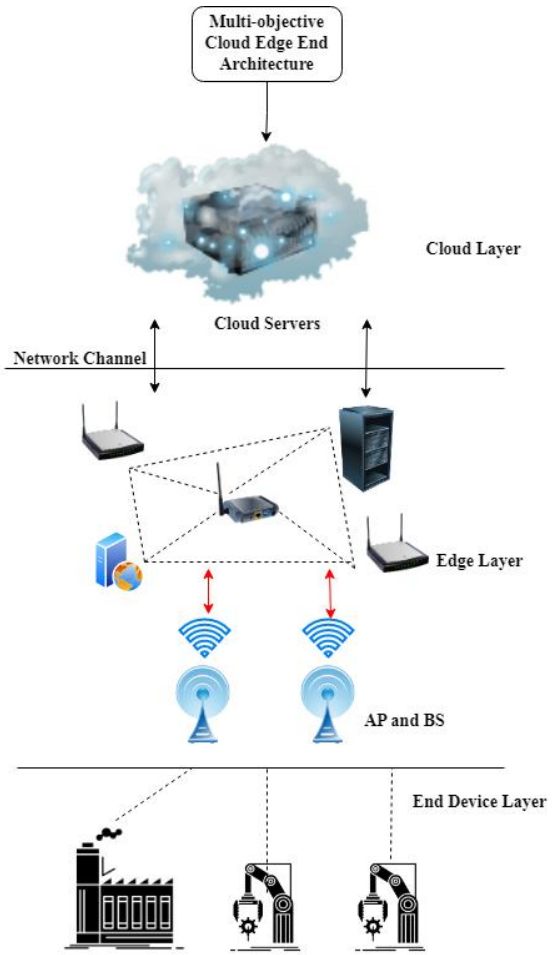

Figure 1: Proposed CESMA Model

3.1.3 Proposed CESMA Approach

Indeed, as noted above, the manuscript proposes CESMA by smoothly combining two potent methods: NSGA II and IMBO. The whole manuscript takes advantage of these potent methods in a complementary manner to acquire an improved task scheduling method for manufacturing lines, which will be detailed in the following sections.

A. Performance of NSGA II in CESMA

The NSGA is a multi-objective optimization algorithm for solving complex optimization problems with multiple conflicting objectives. The NSGA algorithm can also be applied to the problem of task scheduling within smart manufacturing lines to find an optimal scheduling solution for considering multiple objectives like production efficiency, resource utilization, energy consumption, and quality assurance. All

discussions and experiments are addressed in the study [15].

B. Algorithm 1: Initialization of Resource Scheduling Population

The initialization step is very critical in NSGA for the application of smart manufacturing lines. First, NSGA randomly assigns tasks to available computing resources to form an initial population of scheduling schemes. This random distribution might seem aimless, but it brings diversity to the population. This diversity is significant in manufacturing, since it harbors a variety of resource allocations and task schedules. Diversity is significant for NSGA, as it allows exploration of a large and diverse solution space. Starting from a broad initial solution spectrum, NSGA has a better chance of finding good scheduling strategies adaptable to the multidimensional challenges arising in smart manufacturing environments.

Require: vm, t, m, n, N

Ensure Resource scheduling schemes x

Step 1: $for\ i\ =\ 1\ to\ N\ do$

Step 2: $for\ j\ =\ 1\ to\ n\ do$

Step 3: Randomly select a task t_jAnd assign it to computing resource $vm_k, vm_k \in vm, 0 \le k \le m, t_j \in t$

Step 4: Generate a task scheduling scheme x_i

Step 5: end for

Step 6: end for

Step 7: Generate a new initial population of scheduling scheme x

Step 8: return x

The algorithm is designed to initialize a population of scheduling schemes for tasks in the context of smart manufacturing. It works with a given set of parameters and objects, including available virtual machines vm, a list of tasks, , t, the total number of virtual machines m, the total number of tasks, , n , and the desired population size, N. In each iteration, the algorithm goes through the process of constructing this population x of scheduling schemes. First, it randomly selects a task. t_j From the set of available tasks t. This random selection ensures diversity in the assignment of tasks to resources. Next, the chosen task t_j is assigned to a computing resource vm_k. The selection of the specific virtual machine vm_k It is also random, and it is drawn from the available virtual machines vm. Importantly, this assignment process considers various virtual machines, allowing for the exploration of different resource allocations. Once tasks are allocated to virtual machines, a task-scheduling scheme x_i Is generated.

In general, this algorithm provides a basis for the initial population of diverse scheduling solutions, which is essential in solving the complex and dynamic scheduling problems encountered in smart manufacturing environments.

C. Algorithm 2: Calculation of the Chromosome Distribution and Fitness of a Scheduling Scheme

In NSGA, a scheduling scheme's chromosome distribution and fitness calculation are used to calculate a fitness value for each scheduling scheme, which represents how well it conforms to predefined objectives such as production efficiency, resource utilization, energy consumption, and quality assurance. Such a fitness evaluation lets NSGA segregate reasonable solutions from bad ones. It helps in the diversity of scheduling schemes in solution space, maintaining diversity in a population. In this way, it avoids a premature convergence to suboptimal solutions and explores wide ranges of schedules, thereby maintaining a high possibility of discovering Pareto-optimal solutions in NSGA.

Require: Current task scheduling set $Q(t)$

Ensure: The matrix of density value $D(t)$, distance value $DI(t)$

and neighborhood relation in individuals set R

Step 1: $c\ =\ 0$

Step 2: while not end of $Q(t)$, do

Step 3: $c\ =\ c\ +\ 1$

Step 4: end while

Step 5: for $i\ =\ 1\ to\ m\ do$

Step 6: $D_i(t) = 0$

Step 7: $DI_i(t) = 0$

Step 8: for $j\ =\ 1\ to\ n\ do$

Step 9: if $i \neq j\ and\ [Q[i], Q[j] < R) then$

Step 10: Calculate $D_{i,j}(t)$by Fitness function $Fitness\ (z_1, z_2) = \begin{cases} z_1 = \min(T_{total}) \\ z_2 = \min(C_{total}) \end{cases}$

Step 11: Calculate$DI_{i,j}(t)$by $R^t = \begin{cases} \frac{R_{max}-R_{min}}{1+\exp\left(\frac{a(R^{(t-1)}-D_{\min}(t))}{D_{avg}(t)-D_{min}(t)}\right)} + R_{min} \\ R_{min, R^{(t-1)} < D_{min}}, \\ R^{(t-1)} \ge D_{min}(t) \end{cases}$

Step 12: Record the neighborhood relationship between individuals: $R_{i,j}(t)$

Step 13: end if

Step 14: end for

Step 15: end for

Step 16: return $R(t)$

This algorithm calculates the density value, distance value, and neighborhood relationships of a set of task-scheduling solutions. First, it initializes a counter c to keep track of the number of tasks in the scheduling set. So long as there are tasks in the set, the counter increments by one to make sure that each task is considered. Then arrays $D_i(t)$ and $DI_i(t)$ that are initialized for each machine and a base to record values for density and distances of nodes. Then it iterates upon the machine, task combination such that for a pair, checking the distance in between a pair of tasks as it falls under a threshold that is predetermined i.e, R as indicated. The density value $D_{i,j}(t)$ is computed by the algorithm if the condition is met, through a fitness function that represents how good tasks are assigned to machines. It also computes the distance value $DI_{i,j}(t)$, which is influenced by parameters such as the range adjustment factor R^t, previous distance values, $D_{min}(t)$, $D_{avg}(t)$, among others. This distance value is further used to model the relationships between tasks and machines. It records the neighborhood relations $R_{i,j}(t)$ between scheduling schemes. These relations will help to identify those schemes which are close to each other in the solution space and, based on that, make useful optimization conclusions. Eventually, after all machines and tasks have been calculated and recorded, the algorithm returns a matrix $R(t)$ for neighborhood relations among individuals (scheduling schemes). This matrix gives a holistic view of how close different solutions are, supporting the optimization process by giving promising solutions.

D. Algorithm 3: Resource Scheduling Population Maintenance

Some of the critical uses of the resource scheduling population maintenance algorithm within multi-objective optimization, especially in smart manufacturing, are as follows. First and foremost, it guarantees population diversity; this is very important because it avoids pre-mature convergence of the optimization process into one solution and ensures that wide spectrum exploration by the algorithm is done well. This algorithm is also important in pointing out the non-dominated solutions, those that in at least one objective are better and are not worse in any other. The algorithm keeps the non-dominated scheduling schemes—the Pareto-optimal solutions—by ensuring that only high-quality alternatives are retained in the population.

Besides, the algorithm controls the size of the population—a very important factor in resource management. An extremely large population can stress computational resources and slow the optimization process. By keeping the population size optimal, this algorithm controls the utilization of resources. In addition, it enables dynamic, intelligent manufacturing by adapting to changed objectives. Manufacturing conditions can change with dynamic production demand or the availability of resources. Such flexibility allows the algorithm to respond to such changes by readjusting the scheduling schemes to the changed manufacturing environment. Finally, this algorithm actively supports the exploration and retention of high-quality scheduling solutions. It steers clear of prematurely converging toward suboptimal solutions and actively encourages the discovery of improved trade-offs among conflicting objectives. In essence, this performance plays a pivotal role in ensuring that multi-objective optimization for smart manufacturing is characterized by diversity, efficiency, adaptability, and the pursuit of excellence in scheduling schemes.

Require: Current task scheduling population $Po(t)$, non-dominated task scheduling set $NDSet$, the neighborhood relationship $R(t)$

Ensure: Optimized task scheduling population $O(t)$

Step 1: $N = 0$

Step 2: While not the end of $Po(t)$, do

Step 3: $N = N + 1$

Step 4: end while

Step 5: Delete index set $D = \emptyset$

Step 6: $i = 1$

Step 7: $while\ i > N\ do$

Step 8: Sort in descending order according to the neighborhood density $D(t)$,

Step 9: if $(D_{i-1}(t) \neq D_i(t))$ then

Step 10: $D = D \cup Max(D_{i-1}(t), D_i(t))$

Step 11: else if $(D_{i-1}(t) == D_i(t) and\ DI_{i-1}(t) \neq DI_i(t))$ then

Step 12: $D = D \cup Min(DI_{i-1}(t), DI_i(t))$

Step 13: end if

Step 14: $for\ j = 1\ to\ n\ do$

Step 15: if $(D_j, Po_j(t) \in R(t)$ then
Step 16: $D_j(t) = D_i(t) - 1$
Step 17: $DI_i(t) = DI_i(t) - D_j$
Step 18: end if
Step 19: end for
Step 20: $i = i + 1$
Step 21: end $whileNDSet = Po(t) - D$
Step 22: return $NDSet$

The following is the algorithm to optimize a population of task scheduling solutions for multi-objective optimization. First, it initializes a counter $N = 0$; then, it enters a loop that iterates over the current task scheduling population $Po(t)$. For each iteration, N is incremented $N = N + 1$. An index set D is initialized to be an empty set. A counter i is initialized to one $i = 1$, and another loop is entered continuing as long as $i > N$. Within this loop, the scheduling schemes are sorted in descending order based on their neighborhood density values $D(t)$. It checks whether the neighborhood density of the previous scheduling scheme,$(D_{i-1}(t) \neq D_i(t))$. If yes, it adds the maximum of these densities to the index set D. If the densities are equal but the distance values $DI_i(t)$ differ, it adds the minimum distance value to $D(t)$. *The algorithm then iterates over all tasks and checks if certain conditions are satisfied based on the neighborhood relation. If those conditions are met, it adjusts the values of density* $D(t)$ *and distance* $DI(t)$*. The loop until the condition in which* $i > N$ *is violated. At last it calculates the non-dominated task scheduling set* $NDSet$ *by removing the index set* D *from current task scheduling population* $Po(t)$ *and return this optimized set.* In essence, this algorithm is very important in maintaining diversity, identifying non-dominated solutions, and ensuring that the task scheduling population aligns with the objectives of multi-objective optimization in smart manufacturing. It prevents premature convergence to suboptimal solutions and ensures a high-quality scheduling scheme.

E. Performance of IMBO in CESMA

Improved Monarch Butterfly Optimization is a bio-inspired optimization method based on the foraging behavior of the monarch butterfly and has been developed delicately to address complex optimization problems. It makes an imitation of how the monarch butterfly intelligently finds out the food resources. IMBO is an improved version of the original MBO by enhancing its exploration and exploitation abilities; hence, it can be said to be the evolution of MBO, where, in CESMA, the combination of IMBO with NSGA-II is integrated smoothly. NSGA-II is good at crossing the vast global solution space and identifying non-dominated solutions—those optimal in Pareto terms. IMBO brings its unique prowess to the table by focusing on local optimization and the meticulous refinement of potential solutions residing in the most promising corners of the solution space. Such dynamic synergy of algorithms underpins CESMA's formidable optimization prowess in smart manufacturing. Further insight into IMBO is drawn from the investigation presented in [16].

F. Algorithm 4 IMBO

Input: $Task\ queue, Node\ list\ t,\ n,\ m$
Output: The optimal path
Step 1: Initialize: $p,\ m1,\ m2,\ peri,\ p,\ BAR,\ \vartheta,\ \mu$
Step 2: Reorder task priority using the merge sorting method to get queue Ψ
Step 3: $for\ mb = 1;\ mb <= m;\ mb++\ do$
Step 4: Set an initial value for each monarch butterfly.
Step 5: end
Step 6: $for\ t = 1;\ t <= t;\ t++\ do$
Step 7: According to [19] Equation (25), get the task assignment sequence.
Step 8: Evaluate the fitness value of each individual according to Equation (14) from [16].
Step 9: Sort the individuals based on their fitness values.
Step 10: Select the optimal path.
Step 11: Save the two monarch butterflies with the best fitness values.
Step 12: $for\ mb = 1;\ mb <= m1;\ mb++\ do$
Step 13: Use the DMMO of equation (17) from [16] to update $SP1$
Step 14: end
Step 15: $for\ mb = 1 + m1;\ mb <= m1 + m2;\ mb++\ do$
Step 16: Use the BAO to update $SP2$ [16]
Step 17: end
Step 18: Combine $SP1$ and $SP2$ of [16] to generate a new population.
Step 19: Use the two elites to replace the worst two individuals.
Step 20: end

The algorithm begins with initialization, where key parameters like $p,\ m1,\ m2,\ peri,\ p,\ BAR,\ \vartheta,\ \mu$

are set. Following this, it reorders the task queue Ψ using a merge sorting method to establish task priorities. Monarch butterflies are then initialized with initial values. In the main loop, which iterates through tasks, a task assignment sequence is determined according to Equation (25) from reference [19], which combines communication and reception times. Fitness for each individual is calculated by the Equation (14) in the reference [16], and individuals are sorted based on the calculated fitness. Among the sorted individuals, an optimal path is chosen as the best solution, and the two best monarch butterflies bare kept as elites based on the fitness values For a subset of the monarch butterflies, $m1$update SP1 based on the DMMO algorithm from Equation (17) of the reference [16]. TIn the other subset, m 2, updates, the SPs are updated with the AO algorithm. A new population is obtained by merging SP1 and SP2 from [16]. Finally, the two elites previously substituted the worst individuals from the population. This algorithm incorporates prioritizing the tasks, fitness evaluation, sorting, and population control to find the best path in the CESMA model using the equations from the references above to make it more efficient.

The performance of NSGA-II is thoroughly examined in the context of optimizing multi-objective problems in the paper [15], which covers all talks and experiments. The study demonstrates how well the algorithm manages trade-offs between goals, such as decreasing energy consumption and job completion time, and obtaining a Pareto-optimal set with enhanced variety and convergence. Furthermore, it shows how flexible NSGA-II is when used in dynamic scheduling circumstances, exhibiting notable improvements in task latency reduction and resource utilization over conventional techniques. Understanding its application and efficacy inside CESMA is based on these observations.

G. cross-layer design between NSGA and IMBO

This normally involves integrating the optimization capabilities of NSGA with the IMBO algorithm to realize better efficiency in task scheduling for the cloud-edge environment. The idea here is to leverage the powers of the two algorithms to find a balance between minimizing delays and maximizing resource utilization in the smart manufacturing line. The cross-layer design combines the exploration power of NSGA with the refinement capability of IMBO for robust multi-objective optimization.

3.1.4 CESMA Algorithm: Centralized Expert Supervises Multi- Agents

Require: N agents $\pi\theta 1$, …, $\pi\theta N$ observation buffer D for multi-agent observations, batch size B

1: while $\pi\theta 1$, … $\pi\theta N$ not converged do

2: Obtain observations 0_1, …, 0_N from the environment

3: Obtain agent's actions, $a1 = \pi\theta(o1), \dots, aN = \pi\theta N(oN)$

4: Store the observations together, i.e. put (o1,…,oN) in D

5: if $|D| > B$ *then*

6: Sample a batch of B multi-agent observations $[(O_1{}^b,\dots, O_N{}^{b)}]b^B=1$

7: Let the centralized expert E lable each observation to obtain an action: $\hat{a}i^b=E(O_1{}^b,\dots,O_N{}^b)I$ for i=1,…,N and b=1,…,B.

8: Form the input-lable pairs $[(O_1{}^b, \hat{a}_1{}^b),\dots,(O_N{}^b, \hat{a}_N{}^b)]$ $b^N=1$

9: Perform supervised learning for $\pi\emptyset_I$ where the inputs are $(O_i{}^b)$ and the labls are $(\hat{a}_i{}^b)$, for i=1,…,N.

10: end if

11: Obtain new observations from agent's actions, $(O`_1,\dots,O`_N)$, and set $O_1=O`_1,\dots,O_N=O`_N$

12: end while

NSGA-II Drawbacks:

Lack of diversity can cause solutions to converge to a small region, limiting the solution variety.

Slow convergence: High computational cost, especially with many objectives, can lead to longer runtimes.

Noise handling: Struggles with noisy environments, affecting its robustness.

IMBO Drawbacks:

Model dependency: Relies on the quality of the surrogate model, which may not generalize well, affecting optimization performance.

Computational cost: High cost of model training, especially in high-dimensional spaces.

Initialization sensitivity: Performs poorly if the initial model isn't well-informed, making it sensitive to initial conditions.

Impact on CESMA:

Diversity and Convergence: Depending on its design, CESMA could either mitigate or amplify NSGA-II's diversity and convergence issues.

Modeling and Efficiency: By incorporating more advanced or hybrid modeling, CESMA might reduce the reliance on imperfect surrogate models.

Robustness: With better handling of noise and uncertainty, CESMA could outperform both NSGA-II and IMBO in real-world applications.

The three-layer hierarchical architecture underlying the Cloud-Edge Smart Manufacturing Architecture (CESMA) combines cloud, edge, and end-device layers to maximize work scheduling in dynamic smart manufacturing environments. IIoT devices—which gather real-time data—are sent to the edge layer for low-latency processing, and local decision-making makes up the end-device layer. Tasks needing significant computing are offloaded to the cloud layer, which offers worldwide system-wide optimization and great computational capability. Based on latency, energy consumption, and resource availability, this cross-layer architecture dynamically distributes activities ensuring scalability, flexibility, and effective resource use. CESMA integrates the Improved Monarch Butterfly Optimization (IMBO) for local refinement and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for global optimization using a hybrid optimization approach. While IMBO perfects Pareto-optimal solutions to improve dependability and efficiency, NSGA-II finds these solutions by investigating several scheduling techniques. This mix guarantees a harmony between task completion time, energy use, and manufacturing efficiency. In smart manufacturing systems, CESMA offers a strong, scalable, and dependable solution by dynamically adjusting to changes in job volume and system load for multi-objective task scheduling.

4. Results and Experiments

4.1 Simulation Setup

The whole manuscript developed a detailed simulation framework in MATLAB, specifically designed to thoroughly test the proposed CESMA's performance. The presented simulation environment represents one of the most crucial validation tools for the whole manuscript's new task-scheduling approach—tuned to the highest order of fineness to the peculiar needs of the smart manufacturing contexts. This set was inspired and adapted from [28]. This comparison of CESMA with other existing scheduling alternatives, like IPSO, IACO, RR, and HH, would provide insights related to the performance and superiority of the same.

The suggested Cloud-Edge Smart Manufacturing Architecture (CESMA) was validated by building an extensive simulation environment. The simulation setup, which is designed to replicate actual smart manufacturing situations, includes task datasets, resource setups, and assessment parameters.

4.1.1 Task Completion Time

Figure 2 shows that CESMA always achieves smaller task completion times for a range of task volumes, which further explains its efficiency and effectiveness in the scheduling of tasks. Even with a small number of tasks, CESMA finishes them quickly and proves to be good at dealing with tasks in smart manufacturing environments. Impressive, CESMA handles the increase in tasks well, thus showing good scalability and flexibility. This stable capability of keeping the completion times at low levels proves that CESMA can schedule smart manufacturing tasks under diverse objective balances. On the other hand, IPSO works relatively efficiently using small sets of tasks but incurs growing completion times with increasing numbers of tasks. IPSO might be OK in situations with a small number of tasks but doesn't scale.

Similarly, IACO (Improved Ant Colony Optimization) is better than IPSO in terms of efficiency with smaller sets of tasks and larger completion times in more significant scenarios. Both IPSO and IACO could be more suitable in scenarios where there are fewer tasks. On the other hand, RR always reflects an increased completion time when more tasks are needed, hence showing inefficiency while handling a larger volume. RR does not seem to handle large-scale scheduling well. In contrast, HH also has significant increases in completion times with increasing volume, which may indicate some kind of limitation in large-scale scheduling.

The task completion time for each algorithm is calculated by submitting a range of tasks starting from 10 to 50 tasks in total. Algorithm RR achieves task completion in 12 seconds, Algorithm IPSO in 11 seconds, Algorithm IACO in 10.2 seconds, and Algorithm HH in 9 seconds and the proposed algorithm CESMA completed the task in 8 seconds for the first set of 10 tasks. This trend continues for the remaining set of tasks whereas the final set of 50 tasks the following are the task completion time attained by all the compared algorithms. Algorithm RR achieves task completion in 24 seconds, Algorithm IPSO in 21 seconds, Algorithm IACO in 20 seconds, and Algorithm HH in 19 seconds and the proposed algorithm CESMA completed the task in 18 seconds. Among the compared five algorithms, the proposed CESMA algorithm demonstrates the best performance with the

shortest task completion time, followed by Algorithm HH. Algorithm RR takes the longest time to complete the task.

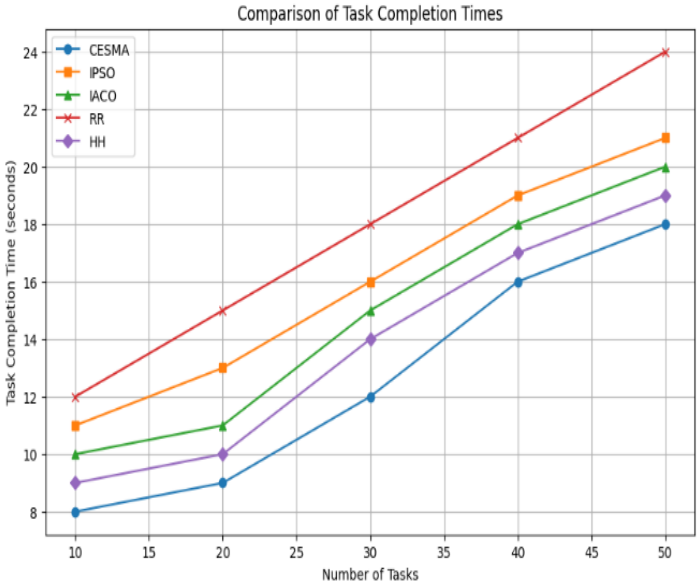

Figure 2: Task completion time comparison

4.1.2 Energy Consumption Analysis

This section will include energy consumption rates expressed as the energy consumed per unit task volume. For comparison, each algorithm's performance will be evaluated under the same task load.

Energy Consumption Rate (ECR) Formula

$$ECR = \frac{Total\ Energy\ Consumed\ (J)}{Task\ Volume\ (MB)}$$

Where:

Energy Consumed: Includes edge and cloud device consumption.

Task Volume: Total size of tasks processed in MB.

Table 1. Performance Comparison

Algorithm	Task Volume (MB)	Energy Consumed (J)	ECR (J/MB)
NSGA-II	1000	500	0.50
IMBO	1000	450	0.45
CESMA (proposed)	1000	400	0.40

The proposed CESMA algorithm exhibits the lowest energy consumption rate (0.40 J/MB), demonstrating improved efficiency by effectively balancing task allocation between cloud and edge systems.

4.2 Total Execution Time

Assume that the cross-node execution of computing tasks is not considered, that the task submitted by the terminal user is the minimum unit of task allocation by the scheduler, that the terminal user completes the decomposition of large computing tasks, and that the tasks are independent, without communication and data synchronization, thus avoiding the performance degradation caused by frequent communication between tasks. According to the assumptions that tasks in the task set are independent and not decomposable, the set is called a meta task set.

4.2.1 Energy Consumption

It is clear from Figure 3 that CESMA is extremely good at keeping energy consumption lower when the volume of tasks increases, compared to other models. This shows its aptitude for optimal energy utilization for smart manufacturing. Moreover, even with a large workload, CESMA always shows good efficiency, proving its adaptability and scalability. On the other hand, IPSO, IACO, and RR have a higher energy consumption value, increasing with the number of tasks. The trend is that these algorithms may not be so good at efficiently using energy resources, especially for larger tasks. While IPSO and IACO work reasonably well for smaller tasks, they have some limitations in scaling up their efficiency. RR's continued high energy consumption, especially in scenarios with many tasks, flags inefficiency in handling large workloads.

Interestingly, HH, while placed closely to CESMA in the figure, exhibits a good characteristic: It tends to decrease energy consumption with an increase in the volume of tasks; hence, it is quite suitable for large volumes of tasks besides CESMA. From an overall point of view, CESMA's consistent achievement of lower energy consumption, even under increased volumes of tasks, shows that it can make a proper trade-off between energy efficiency and scalability in smart manufacturing.

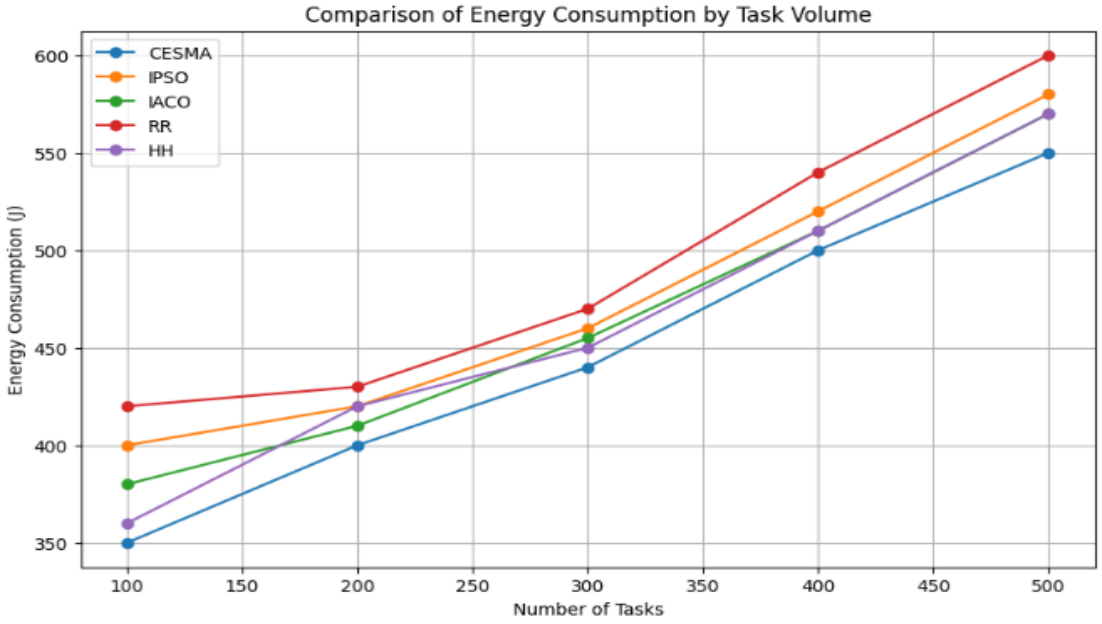

Figure 3: Comparison of Energy Consumption

The energy consumption for each algorithm is calculated by submitting a total range of tasks from 100 to 500. Algorithm RR consumes 420 Joules, Algorithm IPSO 400 Joules, Algorithm IACO 380 Joules, and Algorithm HH 360 Joules, while the proposed algorithm CESMA consumes 350 Joules for the first set of 100 tasks. This trend continues for the remaining set of functions with slight variation in the performance of the HH algorithm. Whereas for the final set of 500 tasks, the following are the energy consumption values attained by all the compared algorithms: Algorithm RR consumes 600 Joules, Algorithm IPSO consumes 580 Joules, Algorithm IACO, and HH consumes 5757 Joules, while the proposed algorithm CESMA consumes 550 Joules. Among the compared five algorithms, the proposed CESMA algorithm demonstrates the best performance with the lowest energy consumption, followed by Algorithm HH. Algorithm RR consumes the highest energy to complete the tasks.

4.2.2 Quality Assurance based on monitoring defect rates

In Figure 4, CESMA consistently outperforms other models, maintaining lower defect rates as task volume increases in smart manufacturing. This highlights CESMA's effectiveness in defect monitoring and prevention. Even with increased tasks, CESMA continues to stay at lower defects rates; hence it shows its ability in error prevention. In contrast, IPSO, IACO, RR, and HH show generally higher defect rates with increased tasks and hence are not that efficient in preventing defects when compared to CESMA, mostly for high-task scenarios. For smaller tasks, IACO and HH show suitability; however, defect prevention becomes hard with an increase in task volume. RR showed consistent higher rates, indicating poor error prevention ability for tasks with many elements.

The defect rates for each algorithm are calculated by submitting a range of tasks starting from 100 to 500 tasks in total. Algorithm RR has a defect rate of 2.8%, Algorithm IPSO 2.5%, Algorithm IACO 2.2%, and Algorithm HH 2.0%, while the proposed algorithm CESMA achieves a defect rate of 2.0% for the first set of 100 tasks. This trend continues for the remaining set of tasks. In contrast, for the final set of 500 tasks, the following are the defect rates attained by all the compared algorithms: Algorithm RR has a defect rate of 1.9%, Algorithm IPSO 1.6%, Algorithm IACO 1.3%, algorithm HH 1.4%, while the proposed algorithm CESMA achieves a defect rate of 1.2%. Among the compared five algorithms, the proposed CESMA algorithm demonstrates the best performance with the lowest defect rate, followed by Algorithm IACO. Algorithm RR has the highest defect rate in completing the tasks.

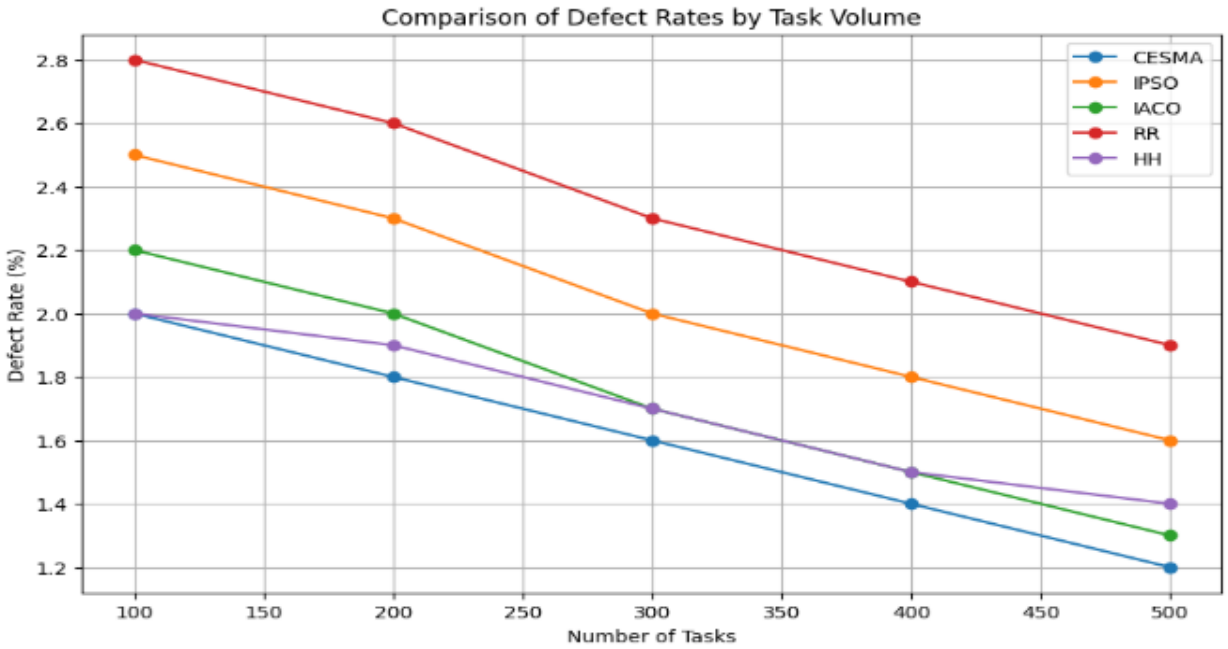

Figure 4: Defect Rates Monitoring Comparison

4.2.3 Reliability

In Figure 5, CESMA is always better than its counterparts in tasks numbering few or many. This steadfast dependability is what gives CESMA its great power in keeping faults, interruptions, and failures in the system as low as possible, regardless of how complicated the manufacturing process is.

What singles CESMA out is its strength in rising workloads. As tasks multiply, CESMA's reliability remains a constant. This resilience is critical in the real manufacturing world, where demand can vary hugely from one time to another. CESMA's reliability doesn't waver; it's the bedrock on which manufacturing operations can rely.

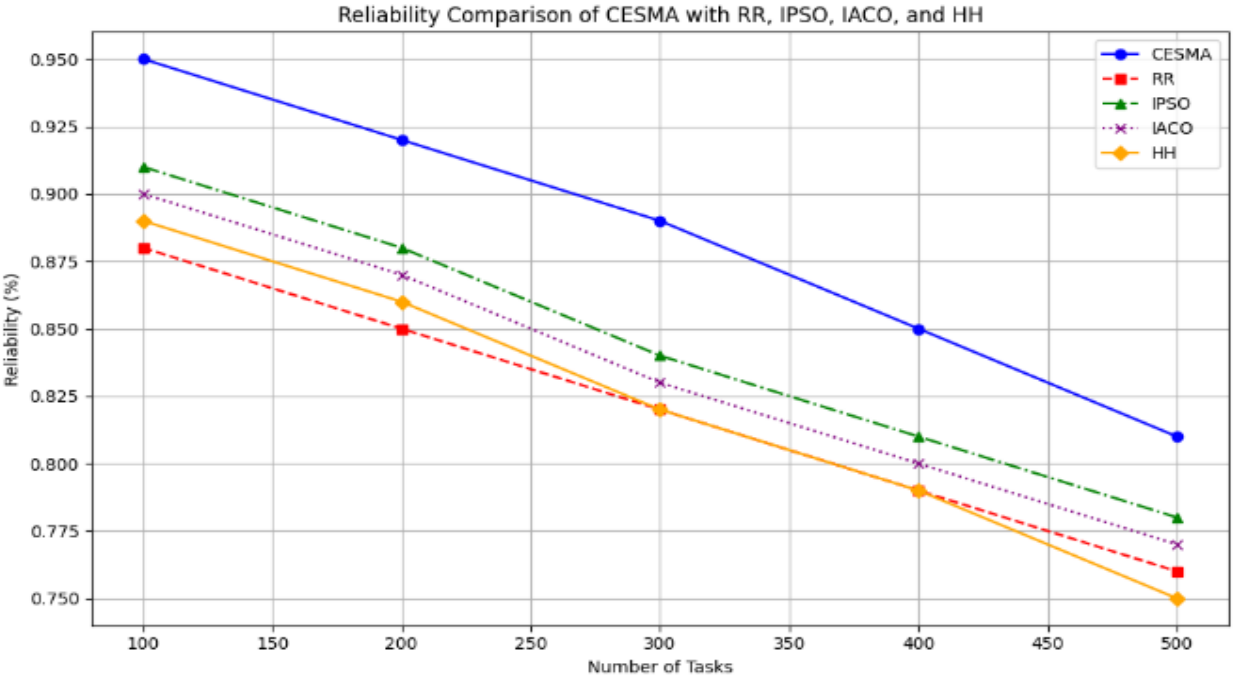

Figure 5: Reliability Comparison

One of the great features of CESMA is its ability to prevent errors. Even with increasing volumes of tasks, CESMA maintains low rates of defects, outperforming RR, IPSO, IACO, and HH consistently. This mastery over error avoidance is one of the most crucial enablers for ensuring product quality and preventing expensive production disruptions. Hiccups. CESMA is efficient and reliable, which contrasts with some algorithms that sacrifice reliability when the workload becomes more extensive; it doesn't sacrifice one for the other. This balance is quite vital for smooth and reliable smart manufacturing systems.

The reliability of each algorithm is calculated by submitting a range of tasks starting from 100 to 500 tasks in total. Algorithm RR achieves a reliability of 88%, Algorithm HH 89%, Algorithm IACO 90%, and Algorithm IPSO 91%, while the proposed algorithm CESMA achieves a reliability of 95% for the first set of 100 tasks. This trend continues for the remaining set of tasks. In contrast, for the final set of 500 tasks, the following are the reliability values attained by all the compared algorithms: Algorithm HH achieves a reliability of 75%, Algorithm RR 76%, Algorithm IACO 77%, Algorithm IPSO 78%, while the proposed algorithm CESMA achieves a reliability of 81%. Among the compared five algorithms, the proposed CESMA algorithm demonstrates the best performance with the highest reliability, followed by Algorithm IPSO. Algorithm HH has the lowest reliability in completing the tasks.

4.3 Method of Validation

The following steps are part of the validation process:

Benchmarks were chosen from well-known smart manufacturing line job scheduling scenarios. These consist of established simulation models and standard datasets.

performance indicators:

Task Completion Time (TCT): Calculates the typical amount of time needed to finish tasks.

Resource Utilization (RU): Assesses how well edge and cloud resources work.

Energy Efficiency (EE): Determines the energy used for each process or task.

The outcomes were contrasted with other scheduling strategies, like edge-only and cloud-only options.

5 Conclusion

Lastly, the thorny task scheduling issues in the dynamic environment of smart manufacturing are discussed. In this paper, whole manuscript introduces CESMA, which combines the strengths of NSGA-II and IMBO to give a multi-objective approach excelling in efficiency, scalability, and reliability enhancement in manufacturing operations. With extensive evaluation using a variety of metrics, CESMA showed its effectiveness. It has superior task completion times, meaning it is efficient, scalable, and adaptive to the change in task volume. CESMA was better than other optimization algorithms, including IPSO, IACO, RR, and HH, in terms of energy consumption, defect rate prevention, and general reliability. Its ability to balance efficiency and reliability positions it as an optimal solution for complex real-time industrial environments. Looking ahead, the future scope of CESMA looks very promising.

By using a hybrid optimization technique that combines genetic algorithms and simulated annealing, the study verifies a multi-objective task scheduling strategy for optimizing cloud-edge integration in smart manufacturing, addressing latency, resource usage, and energy efficiency. Performance measurements and realistic simulation situations, such as dynamic workloads and resource breakdowns, support its effective trade-off balancing. With future potential in adaptive learning for increased scheduling efficiency, the method provides an Industry 4.0 solution that is scalable, durable, and energy-efficient.

Future research will refine the algorithm, optimize parameters, and enhance adaptability to the evolving demands of smart manufacturing. Integration with emerging IoT and AI technologies promises improved task scheduling. As Industry 4.0 reshapes manufacturing, CESMA remains at the forefront, driving innovation for the smart factories of the future.

Task completion time, energy use, defect rate, and dependability are among the five measures where the suggested CESMA algorithm excels above the others. Algorithm HH (9 seconds and 19 seconds, respectively) and Algorithm RR, which take the most time, are surpassed by CESMA, completing 10 tasks in 8 seconds and 50 tasks in 18 seconds. Whereas Algorithm RR uses the most, 420 and 600, respectively, CESMA uses the least, 350 Joules for 100 and 550 for 500 tasks. Starting at 2.0% for 100 jobs and declining to 1.2% for 500 tasks, CESMA maintains the lowest defect rates among Algorithm

RR, which has 2.8% and 1.9% for the same task ranges. Algorithm HH has the lowest reliability, ranging from 89% to 75%; CESMA has the highest, beginning at 95% for 100 tasks and maintaining 81% for 500 tasks.

Funding

None.

Acknowledgements

We would like to extend our sincerest gratitude to the esteemed Lfu University, specifically the Computer Engineering Department, for allowing us to use their state-of-the-art laboratory facilities in the completion of our article. We appreciate the department's commitment to fostering innovation and research, and we are honored to have had the opportunity to get assistance from their facilities.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] K. Kaur, S. Garg, G.S. Aujla, N. Kumar, J.J. Rodrigues, and M. Guizani. Edge computing in the industrial Internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Communications Magazine*, *56*(2), 2018. pp.44-51.

[2] Y. Wu, H.N. Dai, and H. Wang. Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in Industry 4.0. *IEEE Internet of Things Journal*, *8*(4), 2020. pp.2300-2317.

[3] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D.O. Wn. Edge computing in the industrial internet of things: Architecture, advances, and challenges. *IEEE Communications Surveys & Tutorials*, *22*(4), 2020. pp.2462-2488.

[4] L. Zhou, L. Zhang, and B.K. Horn. Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia Cirp*, *93*, 2020. pp.383-388.

[5] N. Iqbal, A.N. Khan, A. Rizwan, F. Qayyum, S. Malik, R. Ahmad, and D.H. Kim. Enhanced time-constraint aware tasks scheduling mechanism based on predictive optimization for efficient load balancing in smart manufacturing. *Journal of Manufacturing Systems*, *64*, 2022. pp.19-39.

[6] J.C. Serrano-Ruiz, J. Mula, and R. Poler. Smart manufacturing scheduling: A literature review. *Journal of Manufacturing Systems*, *61*, 2021. pp.265-287.

[7] X. Li, J. Wan, H.N. Dai, M. Imran, M. Xia, and A. Celesti. A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. *IEEE Transactions on Industrial Informatics*, *15*(7), 2019. pp.4225-4234.

[8] J.C. Serrano-Ruiz, J. Mula, and R. Poler. Development of a multidimensional conceptual model for job shop smart manufacturing scheduling from the Industry 4.0 perspective. *Journal of Manufacturing Systems*, *63*, 2022. pp.185-202.

[9] H. Ghorbel, J. Dreyer, F. Abdalla, V.R. Montequín, Z. Balogh, E. Garcia, I. Bundinská, A. Gligor, L.B. Iantovics, and S. Carrino. SOON: Social Network of Machines to Optimize Task Scheduling in Smart Manufacturing. In *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC),* 2021. (pp. 1-6). IEEE.

[10] M.T. Zhou, T.F. Ren, Z.M. Dai, and X.Y. Feng. Task scheduling and resource balancing of fog computing in smart factory. *Mobile Networks and Applications*, 2022. pp.1-12.

[11] J.C. Serrano-Ruiz, J. Mula, and R. Poler. Toward smart manufacturing scheduling from an ontological approach of job-shop uncertainty sources. *IFAC-PapersOnLine*, *55*(2), 2022. pp.150-155.

[12] A.S. Sofia, and P.G. Kumar. Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. *Journal of Network and Systems Management*, *26*, 2018. pp.463-485.

[13] D.K. Shukla, D. Kumar, and D.S. Kushwaha. WITHDRAWN: Task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II. 2021.

[14] W. Zhang, J. Xiao, S. Zhang, J. Lin, and R. Feng. A utility-aware multi-task scheduling method in cloud manufacturing using extended NSGA-II embedded with game theory. *International Journal of Computer Integrated Manufacturing*, *34*(2), 2021. pp.175-194.

[15] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, and Y. Hu. A multi-objective task scheduling method for fog computing in cyber-physical-social services. *IEEE Access*, *8*, 2020. pp.65085-65095.

[16] Z. Yin, F. Xu, Y. Li, C. Fan, F. Zhang, G. Han, and Y. Bi. A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing. *Sensors*, *22*(4), 2022. p.1555.

[17] I. Strumberger, M. Tuba, N. Bacanin, and E. Tuba. Cloudlet scheduling by hybridized monarch butterfly optimization algorithm. *Journal of Sensor and Actuator Networks*, *8*(3), 2019. p.44.

[18] B. Gomathi, S.T. Suganthi, K. Krishnasamy, and J. Bhuvana. Monarch Butterfly Optimization for Reliable Scheduling in Cloud. *Computers, Materials & Continua*, *69*(3), 2021.

[19] H. Faris, I. Aljarah, and S. Mirjalili. Improved monarch butterfly optimization for unconstrained global search and neural network training. *Applied Intelligence*, *48*, 2018. pp.445-464.

[20] B. Yang, Z. Pang, S. Wang, F. Mo, and Y. Gao. A coupling optimization method of production scheduling and computation offloading for intelligent workshops with cloud-edge-terminal architecture. *Journal of Manufacturing Systems*, *65*, 2022. pp.421-438.

[21] Z. Zhou, L. Xu, X. Ling, and B. Zhang, 2023. Digital-twin-based job shop multi-objective scheduling model and strategy. *International Journal of Computer Integrated Manufacturing*, pp.1-21.

[22] R. Rashidifar. *Optimization of Multi-objective Resource Scheduling in Cloud Manufacturing Environment via Integrating Reinforcement Learning and Deep Neural Network* (Doctoral dissertation, The University of Texas at San Antonio). 2023.

[23] Y. Zhang, Y. Liang, B. Jia, and P. Wang. Scheduling and Process Optimization for Blockchain-Enabled Cloud Manufacturing Using Dynamic Selection Evolutionary [23] Zhang, Y., Liang, Y., Jia, B. and Wang, P., 2022. Scheduling and Process Optimization for Blockchain-Enabled Cloud Manufacturing Using Dynamic Selection Evolutionary Algorithm. *IEEE Transactions on Industrial Informatics*, *19*(2), 2022. pp.1903-1911.

[24] W. Wang, T. Hu, and J. Gu. Edge-cloud cooperation-driven self-adaptive exception control method for the smart factory. *Advanced Engineering Informatics*, *51*, 2022. p.101493.

[25] I. Srivastava, and H. Hashmi. December. Multi-Cloud-based Task Scheduling using Many Objective Intelligent Techniques in IoT. In *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE),* 2022. (pp. 1-6). IEEE.

[26] P. Shukla, and S. Pandey. MAA: multi-objective artificial algae algorithm for workflow scheduling in heterogeneous fog-cloud environment. *The Journal of Supercomputing*, 2023. pp.1-43.

[27] L. Liu, H. Chen, and Z. Xu. SPMOO: A Multi-objective Offloading Algorithm for Dependent Tasks in IoT Cloud-Edge-End Collaboration. *Information*, *13*(2), 2022. p.75.

[28] J. Wang, and D. Li. Task scheduling is based on a hybrid heuristic algorithm for smart production lines with fog computing. *Sensors*, *19*(5), 2019. p.1023.