2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 5-8, 2025. Vienna, Austria

Latency-minimized Computation Offloading in 3C Manufacturing Workshops

Yanan Liu[1], Jing Bi[1], Ziqi Wang[2], Junqi Zhang[1], Haitao Yuan[3] and Jia Zhang[4]

[1]College of Computer Science, Beijing University of Technology, Beijing 100124, China
[2]School of Software Technology, Zhejiang University, Ningbo 315100, China
[3]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[4]Dept. of Computer Science in Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

***Abstract*—With the rapid advancement and integration of Internet of Things technology into manufacturing, industrial workshops in computer, communication, and consumer electronics (3C) manufacturing are increasingly confronted with complex computational tasks during production. However, the limited hardware resources and computational capabilities of local devices often hinder efficient task execution. Computational offloading offers a viable solution by allowing complex computational tasks to be processed on either edge or cloud servers, enhancing the efficiency of computational task handling in production environments. A critical challenge lies in optimizing task offloading among local devices, edge servers, and cloud servers to maximize production efficiency while ensuring reasonable task scheduling. To address this challenge, this work proposes a flexible computational offloading strategy based on an edge-cloud architecture in a smartphone manufacturing workshop. First, a framework for edge-cloud workshop manufacturing is constructed, integrating various smartphone production devices. Based on the edge-cloud framework, a constrained optimization problem for computation offloading is formulated, using latency as the objective in industrial production settings. A time consumption model is employed to optimize computational time, and a novel scheduling strategy named Ivy-Genetic Evolution Algorithm (IGEA) is designed to solve the scheduling problem. The IGEA integrates genetic operators into the Ivy Algorithm to introduce a randomness strategy. Experimental results demonstrate that IGEA significantly outperforms state-of-the-art approaches in optimizing production efficiency.**

***Index Terms*—Internet of things, edge-cloud computing, intelligent optimization algorithm, flexible workshop, and mobile phone manufacturing.**

I. INTRODUCTION

With rapid technological advances and market changes, the integration of information technology with physical systems has become a key driver for manufacturing development. As a core component, the Internet of Things (IoT) is reshaping traditional manufacturing [1], enabling data-driven management and automation [2]. In recent years, the proliferation of IoT devices and improved computing capabilities have continuously improved intelligence in the computer, communication, and consumer electronics (3C) manufacturing sector [3]. Especially in smartphone manufacturing, IoT devices connect to production equipment to collect real-time data, optimizing production stages, and improving efficiency. However, intelligent production also leads to a surge in complex computational tasks. Due to the limited processing power of local industrial devices, executing these tasks is challenging and time consuming.

Computational offloading provides a feasible solution to this problem [4]. In industrial production, edge servers are deployed within the workshop, while long-term external connections with the cloud enable the optimization of computation offloading tasks. By offloading computational tasks arising during production to edge or cloud servers with greater processing power, complex computations can be handled efficiently [5]. However, the process of offloading tasks to the edge or cloud also introduces additional latency due to data transmission. Current research has proposed many algorithms for optimizing different targets in the edge-cloud environment [6]–[10]. Unfortunately, they also exhibit certain shortcomings in practical applications. For example, when facing large-scale and complex tasks, they tend to get trapped in local optima and fail to guarantee global optimality. Additionally, due to issues with parameter tuning and convergence speed, some strategies do not perform stably in production environments with high real-time requirements.

According to the preceding analysis, to further reduce the possibility of falling into local optima and to balance the computing time and transmission latency in the industrial production process, this work proposes a computational offloading strategy for the 3C smartphone manufacturing industry supported by an edge-cloud system. An edge-cloud architecture is constructed that integrates various smartphone production devices. Based on this framework, a constrained optimization problem in industrial production is formulated and solved by a novel intelligent optimization algorithm named Ivy-Genetic Evolution Algorithm (IGEA). It integrates genetic mutation to explore a broader range of potential solutions. While improving production efficiency, IGEA also considers the priority of tasks. Experimental results show that IGEA achieves significantly higher production efficiency, providing a feasible solution for intelligent computational offloading in 3C manufacturing.

The remaining sections of this work are organized as follows. Section II introduces the framework and problem

This work was supported by the Beijing Natural Science Foundation under Grants L233005 and 4232049, the National Natural Science Foundation of China under Grants 62473014 and 62173013, and in part by Beihang World TOP University Cooperation Program (Corresponding author: Junqi Zhang.)

formulation. Section III gives details of IGEA. Section IV presents the performance evaluation, and Section V draws the conclusion.

II. Problem Formulation

Under the edge-cloud architecture shown in Fig. 1, the production system of a smartphone manufacturing workshop comprises multiple tasks and machines. In this architecture, T_i denotes task i, and M_j denotes machine j. Each task typically includes multiple operations, and operation k of task i is denoted as O_{ik}. Production scheduling involves assigning operations to suitable machines to ensure smooth task execution. For example, smartphone coating consists of alignment, recognition, painting, and inspection, each handled by available machines. Some operations require significant computational resources, making computational offloading crucial for overall efficiency. In addition, tasks have varying priorities that are considered during scheduling. These designs improve machine utilization and ensure efficient processing of computational tasks, thus improving overall production performance.

The network integrates 5G, optical fiber, Time-Sensitive Networking (TSN) [11], and Local Area Networks (LAN) [12]. The cloud and edge are connected via 5G, enabling remote data transmission and processing [13]. Within workshops, TSN and LAN ensure real-time, deterministic communication for precise industrial control. Inter-workshop connectivity is achieved through optical fiber, supporting high-speed data sharing and collaboration.

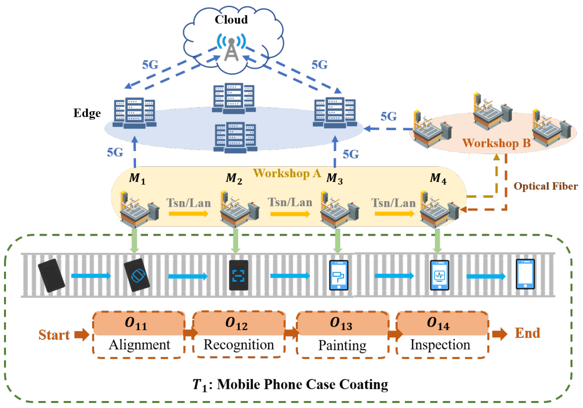

Fig. 1. Edge-cloud system and network architecture in 3C workshop.

A. Time Consumption Model

The time consumption model comprises two primary aspects: the execution time of production tasks and the computation time of computational tasks. The execution time of all production tasks is denoted as T^{α}. It refers to the total time required to complete all production tasks, *i.e.*,

$$T^{\alpha}=\max(t_1^c, ...t_i^c) \tag{1}$$

TABLE I
MAIN PARAMETERS

Notation	**Definition**
T_i	Production task i
M_j	Machine j
O_{ik}	Operation k of T_i
t_i^c	Completion time of T_i
t_i^w	Waiting time for T_i
t_i^p	Processing time for T_i (including all operations)
t_{ik}^o	Processing time for operation k in task i
T^{α}	Execution time of all production tasks
t_i^d	Data transmission latency for T_i
t_i^e	Computation time for T_i
d_i	Data size required by T_i
s^e	Data transfer speed of the edge device
s^c	Data transfer speed of the cloud
c_i	Computational load of T_i
f_j^m	Computational capacity of M_j
f^e	Computational capacity of the edge device
f^c	Computational capacity of the cloud
γ_i	Importance score of T_i
h_i^m	Importance weight of T_i
δ_i	Urgency score of T_i
z_i	Deadline of T_i
ϵ_i	Resource demand index of T_i
$\check{t}^c$	Minimum computational resource value
$\hat{t}^c$	Maximum computational resource value
S^p	Reverse priority score

where t_i^c denotes the completion time of the production task i. The completion time of each task consists of its waiting time t_i^w and processing time t_i^p, *i.e.*,

$$t_i^c=t_i^w+t_i^p \tag{2}$$

$$t_i^p=\sum_{k=1}^{K} t_k^o \tag{3}$$

where t_i^p is the total processing time of all K operations in task i and t_k^o denotes the processing time of operation k.

The computation time of a task refers to the total time required for its execution, considering both data transmission latency t_i^d and computation time t_i^e, *i.e.*,

$$T^{\beta}=\sum_{i=1}^{I} t_i^d+\sum_{i=1}^{I} t_i^e \tag{4}$$

where T^{β} denotes the total execution time of the computational task. In the edge-cloud architecture, the data latency depends on whether the task is executed on a local machine, an edge device, or in the cloud, *i.e.*,

$$t_i^d=\begin{cases} 0 & \text{local devices} \\ d_i/s^e & \text{edge server} \\ d_i/s^c & \text{cloud server} \end{cases} \tag{5}$$

where d_i is the amount of processed data required by task i, s^e is the data transfer speed of the edge device, and s^c is the data transfer speed of the cloud. For locally executed tasks, no data transmission is required. However, when the task is transmitted to the edge or cloud, it needs to consider the upload latency using (5).

The computation time for a task depends on the available computational resources. When the task is executed on a local machine, t_i^e is obtained as:

$$t_i^e = \begin{cases} c_i/f_j^m & \text{local devices} \\ c_i/f^e & \text{edge server} \\ c_i/f^c & \text{cloud server} \end{cases} \tag{6}$$

where c_i is the computational load of the task i, f_j^m denotes the computational capacity of the machine j, f^e donates the computational capacity of the edge server, and f^c denotes the computational capacity of the cloud server.

B. Scheduling Model

Task scheduling is a critical optimization problem in manufacturing. Different production tasks vary in importance, urgency, and resource demand. When handling multiple tasks, determining the optimal execution sequence is key to minimizing production time while accounting for task priority.

1) Task Importance and Task Urgency: The importance of a task is one of the critical factors influencing the determination of its execution priority. Tasks with greater importance should be prioritized. The importance score γ_i is obtained as:

$$\gamma_i = t_i^w \times h_i^m \tag{7}$$

where h_i^m denotes the task's importance weight for task i. (7) shows that tasks with higher weights and waiting longer have higher priorities. Moreover, tasks with stricter time constraints should be prioritized to guarantee timely completion. The urgency score δ_i is obtained as:

$$\delta_i = \exp\left(-(z_i - t_i^w) \times h_i^m\right) \tag{8}$$

where z_i denotes the deadline for task i. (8) demonstrates that as a task approaches its deadline, its urgency increases.

2) Resource Demand Index: Different tasks have different demands for computational resources. To optimize resource allocation, we define a resource demand index and prioritize tasks with lower resource requirements to reduce waiting time and enhance system flexibility. The resource demand index ϵ_i is obtained as:

$$\epsilon_i = \frac{t_i^c - \check{t}^c}{\hat{t}^c - t_i^c} \tag{9}$$

where $\check{t}^c$ and $\hat{t}^c$ denote the minimum and maximum computational resource values, respectively. An increase in demand leads to a higher index value.

3) Reverse Priority Score: Considering task importance, urgency, and resource demand, the reverse priority score S^p of task i is obtained as:

$$S_i^p = \gamma_i \times \delta_i \times \epsilon_i \tag{10}$$

A lower reverse priority score indicates a higher priority for task i. By adjusting the execution sequence of tasks in the scheduling process, it can ensure both production efficiency and personalized production requirements.

C. Objective Function

The Metric R is a comprehensive quantitative metric that integrates several key factors to optimize system performance. These factors include the time consumption of production, data transition, and computations of tasks, and the priorities of different tasks. By weighing these factors in an integrated manner, R can effectively reflect the overall operational efficiency and resource utilization of the system. It is obtained as:

$$R = T^\alpha + T^\beta + S^p \tag{11}$$

In summary, the objective function is to minimize (11), thereby optimizing task scheduling and enhancing overall production efficiency. To optimize the objective function and thereby identify the most efficient production solution in a 3C manufacturing workshop, IGEA is proposed. The next section introduces the implementation details of it.

III. Ivy-Genetic Evolution Algorithm (IGEA)

IGEA improves upon Ivy Algorithm (IVYA) [14] by embedding a genetic mutation process. It combines the global search advantages of the IVYA with the evolutionary mechanism [15], aiming to adaptively adjust the exploration of the search space. IGEA can effectively avoid the local optimum dilemma and enhance global search capability.

Algorithm 1 IGEA

```
Input: $\check{I}$, $\hat{I}$, $\alpha$, $\beta$
Output: Best Fitness $\min(C_b)$
 1: Initialize population $I_1, ...I_i$ with (12).
 2: Calculate growth vector $g_1^v, ...g_i^v$ with (13).
 3: for $it = 1$ to Max Iteration do
 4:     Initialize new population.
 5:     for $i$ in population do
 6:         Select neighbor $f$, calculate $\mu$.
 7:         if $C_i < \mu \times C_b$ then
 8:             Update $I_i$ (neighbor-based) with (16).
 9:         else
10:             Update $I_i$ (global-best-based) with (17).
11:         end if
12:         Update $g_i^v$ with (18)
13:         Crossover and Mutation :
14:         Use $I_i$ as the first parent and select the second parent $I^p$
            randomly.
15:         Perform crossover at $c = \text{randi}(1, D)$.
16:         Generate $I_i^c$ with (19).
17:         if $it \bmod 50 = 0$ then
18:             Generate $P^m$ with (20) and update $I_i^c$ with (22).
19:         else
20:             Generate $P^m$ with (21) and update $I_i^c$ with (22).
21:         end if
22:         Update fitness $C_i$ with (23).
23:     end for
24:     Insert some rand positions with (12).
25:     Merge and sort populations by fitness.
26:     Select $C_b$ with (24).
27: end for
```

A. Initialization

The initialization phase of IGEA involves generating an initial population, where each individual's position in the solution space is randomly selected, *i.e.*,

$$I_i = \check{I} + \mathrm{rand}(1, D) \odot (\hat{I} - \check{I}) \tag{12}$$

where I_i denotes the position of the individual i, which is randomly generated within the search space boundaries defined by the minimum $\check{I}$ and maximum $\hat{I}$. Specifically, the position is determined by a random vector $\mathrm{rand}(1, D)$ with dimension D, ranging from [0,1]. To ensure that the generated position stays within the valid range of the search space, the Hadamard product $\odot$ is used. Each individual is assigned both a growth vector and a fitness value. The growth vector g_i^v is obtained as:

$$g_i^v = \frac{I_i}{\hat{I} - \check{I}} \tag{13}$$

The growth vector is dynamically updated based on the individual's position, reflecting the evolution process in the search space. It indicates the relative change in the position of the individual within the search space. Then, the fitness value of each C_i is obtained as:

$$C_i = f_o(I_i) \tag{14}$$

where f_o denotes the objective function. f_o is obtained as 11, which represents the quality of the solution. A lower fitness value indicates a better solution. After the initialization phase, IGEA proceeds to the population iteration stage.

B. Population Iteration

At the beginning of each generation's main loop, IGEA follows IVYA. It iterates through each individual in the current population, updating its position and evaluating its fitness. When updating positions, each individual compares with the best individual and adjust its position. Interactions between individuals enhance information exchange, enabling effective exploration in the neighborhood. Based on this, IGEA calculates a random coefficient μ for each individual, *i.e.*,

$$\mu = 1 + 2 \times \mathrm{rand}() \tag{15}$$

where μ denotes the ratio coefficient for updating an individual's position and introduces randomness into the search process. Then, the position update depends on the comparison between the current individual's fitness and the fitness of its neighbor. Specifically, if $C_i < \mu \times C_b$, the new position I_i is obtained as (16). Otherwise, the position is updated based on the best individual I_b, using (17).

$$I_i = I_i + |N(1, D)| \odot (I_\mathrm{j} - I_i) + N(1, D) \odot \Delta g_i^v \tag{16}$$

$$I_i = I_b \odot (\mathrm{rand}(1, D) + N(1, D) \odot \Delta g_i^v) \tag{17}$$

where C_b denotes the best fitness value. After each position update, the growth vector is recalculated to ensure that it reflects the latest evolutionary trend of the individual within the search space, *i.e.*,

$$\Delta g_i^v(t+1) = rand^2 \odot (N(1, D) \odot \Delta g_i^v(t)) \tag{18}$$

Building on the population iteration, IGEA further introduces crossover and mutation operations.

C. Crossover and Mutation

The introduction of the genetic operator is an innovation in IGEA. Each undergoes genetic mutation after population updating, which can further enhance population diversity and ensure the algorithm's effective exploration of optimal solutions within the search space. For the crossover, IGEA chooses I_i as the first parent and selects the second parent I^p randomly. Then, it performs a crossover to generate offspring in each iteration. The crossover point is randomly selected, and the portions of the parent individuals before and after the crossover point are combined to form new offspring, *i.e.*,

$$I_i^c = [I_i(1:c), I^p(c+1:D)] \tag{19}$$

where I_i^c denotes the child and c denotes crossover points.

For mutation, the offspring undergoes a mutation operation. The key to mutation is the random selection of mutation positions, followed by the generation of new values for these positions. The formula to get the mutation position is obtained as:

$$p^m = \mathrm{randi}([1, D], 1, \alpha \cdot D) \tag{20}$$

$$p^m = \mathrm{randi}([1, D], 1, \beta \cdot D) \tag{21}$$

where p^m denotes the mutation positions, α and β denote the proportion of mutation position. α and β are key parameters controlling mutation strength. A smaller β is used for local mutation and fine exploration, refining solutions near promising areas. Every 50 iterations, a larger α is applied to escape local optima, enhance global exploration, and prevent premature convergence. Then, the individual can be updated using p^m, *i.e.*,

$$I_i^c(p^m) = \check{I} + (\hat{I} - \check{I}) \times \mathrm{rand}(1, \mathrm{length}(p^m)) \tag{22}$$

After mutation, the system recalculates individual's fitness and selects the best one between I_i^c and I_i using (23).

$$C_i = \min(f_o(I_i^c), f_o(I_i)) \tag{23}$$

D. Random Injection and Fitness Tracking

To enhance population diversity, IGEA periodically introduces I_r of randomly generated individuals using (12). These random individuals help prevent it from converging too early and enhance the coverage of the search space. After each generation, the best fitness of the current individual is recorded, and relevant information is output, *i.e.*,

$$C_b = \min(\mathrm{C}) \tag{24}$$

By tracking the changes in the best solution, IGEA gradually approaches the global optimum. Through the embedding of genetic operations, IGEA achieves a more diversified exploration of the search space based on IVYA. Crossover and mutation operations effectively increase the diversity of the population, while the proportion of mutations α and β control the strength of the mutation, further enhancing its global search capability. Finally, the details of IGEA are shown in the algorithm 1.

IV. Experimental Results and Discussion

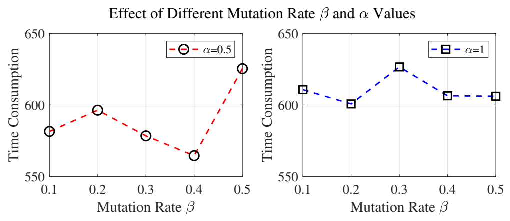

Fig. 2. Mutation Rate Parameters α and β.

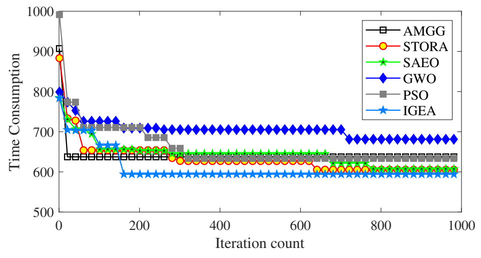

Fig. 3. Time Consumption in each iteration for each algorithm.

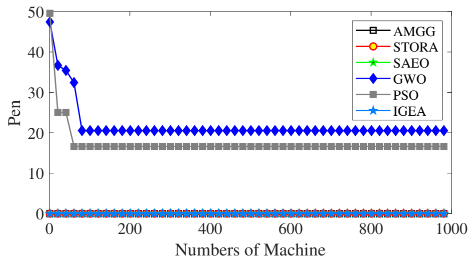

Fig. 4. Penalty in each iteration for each algorithm.

Experiments are conducted to simulate the real situation of computational offloading in a smartphone manufacturing workshop under an edge-cloud architecture. The specific parameter settings are shown in Table II, while the parameter settings of α and β for IGEA are illustrated in Fig. 2. It reveals optimal performance when α is 0.5 and β is 0.4. Then, we compare IGEA with five benchmark algorithms, including GWO [16], PSO [17], STORA [18], AMGG [19] and SAEO [20]. Fig. 3 shows the time consumed by those six algorithms in each iteration under the scenario with 100 machines, 20 tasks, and each task comprising 3 operations. The results show that after 1000 iterations, IGEA consumes significantly less time than the other algorithms. Fig. 4 illustrates the penalty values of the six algorithms under varying numbers of machines, where the penalty value of IGEA is zero. This indicates that IGEA can find effective solutions. In contrast, PSO and GWO perform poorly with higher penalty values, proving that they fail to meet all the constraints.

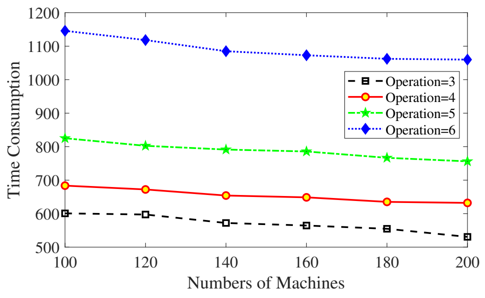

Fig. 5. Cost v.s. O and M for IGEA.

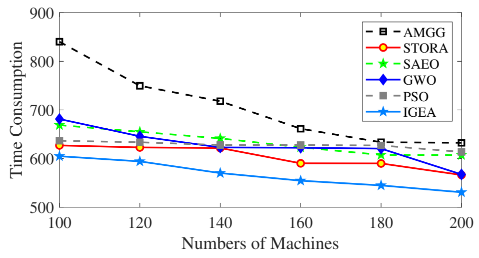

Fig. 6. Time Consumption v.s. M for each algorithm.

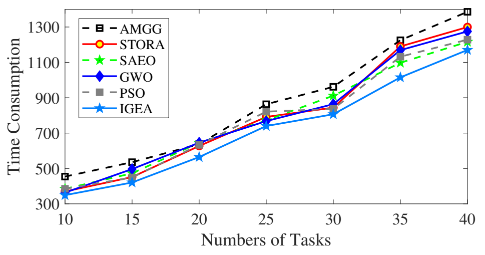

Fig. 7. Cost v.s. T for each algorithm.

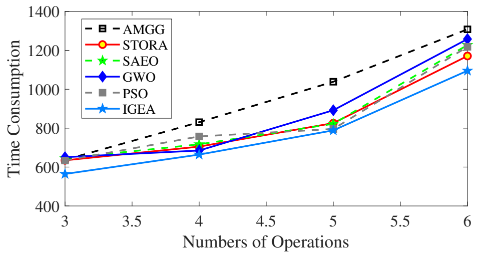

Fig. 8. Cost v.s. O for each algorithm.

Fig. 5 shows the impact of changes in the number of machines and task operations on time consumption. The results indicate that as the number of machines increases, the number of idle machines also increases, thereby reducing the waiting time required for task execution and consequently lowering the time consumption. However, as the number of operations increases, the total number of operations to be processed also

TABLE II
PARAMETER SETTINGS FOR ALGORITHM CONFIGURATION

M	T	O	t^o (ms)	z (ms)	t^c (ms)	t^d (mb)	h^m	f^m (GHz)	f^e (GHz)	f^c (GHz)	s^e (Mbps)	s^c (Mbps)
100	20	3	[40, 180]	[400, 3000]	[100, 1000]	[100, 1000]	[0, 1]	[10, 100]	2×10^4	2×10^6	10^3	10^2

rises, leading to an increase in time consumption. Fig. 6-8 show the time consumed by the six algorithms under different numbers of machines, tasks, and operations, respectively. The experimental results reveal that IGEA consistently maintains the best performance in all scenarios.

V. CONCLUSIONS

With the rapid development of Internet of Things (IoT) technology, its applications in computer, communication, and consumer electronics (3C) manufacturing have expanded significantly, driving transformative changes in the industry. However, the large volume of complex computational tasks generated by IoT devices during production poses major challenges for local industrial systems. To address this, this work proposes a computation offloading strategy for manufacturing workshops based on an edge-cloud architecture.A multi-device edge-cloud framework integrating various smartphone manufacturing equipment is first established. Within this framework, an efficiency-oriented computation offloading and task constraint optimization model is developed under an industrial setting. By incorporating a time consumption model and a scheduling model, the execution order of tasks is adjusted according to their priorities to maximize production efficiency.Furthermore, a novel intelligent optimization algorithm, the Ivy-Genetic Evolution Algorithm (IGEA), is introduced. By embedding genetic operators, IGEA enhances randomness and global search capability. Experimental results demonstrate that IGEA significantly outperforms state-of-the-art algorithms in optimizing production efficiency.

Future work will focus on integrating optimal edge selection strategies and deep learning-based search mechanisms into IGEA, enabling more intelligent and automated production processes. In addition, the adaptability and robustness of IGEA across diverse manufacturing environments and conditions will be evaluated to ensure stable performance in various application scenarios.

REFERENCES

[1] Y. Liu, L. Jiang, Q. Qi and S. Xie, "Energy-Efficient Space–Air–Ground Integrated Edge Computing for Internet of Remote Things: A Federated DRL Approach," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 4845–4856, Mar 15, 2023.

[2] A. Bozzi, S. Graffione, J. -F. Jimenez, R. Sacile and E. Zero, "A Platoon-Based Approach for AGV Scheduling and Trajectory Planning in Fully Automated Production Systems," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 1, pp. 594–603, Jan. 2025.

[3] P. Goswami, R. Hazra, M. Prokysek, K. Choi and M. Eider, "A Novel Approach of Efficient Resource Allocation in Smart Consumer Electronics Device Network," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 3, pp. 5895–5902, Aug. 2024.

[4] H. Zhao, Z. Wang, G. Cheng, W. Qian, P. Chen, J. Yin, S. Dustdar, and S. Deng, "Online Workload Scheduling for Social Welfare Maximization in the Computing Continuum," *IEEE Transactions on Services Computing,* doi: 10.1109/TSC.2025.3570845.

[5] H. Yuan, Q. Hu, M. Wang, S. Wang, J. Bi, *et al.*, "Data-Filtered Prediction With Decomposition and Amplitude-Aware Permutation Entropy for Workload and Resource Utilization in Cloud Data Centers," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 19189–19201, Jun. 2025.

[6] H. Yuan, J. Bi, Z. Wang, J. Yang, and Jia Zhang, "Partial and Cost-minimized Computation Offloading in Hybrid Edge and Cloud Systems," *Expert Systems with Applications*, vol. 250, no. 15, pp. 1–13, Sept. 2024.

[7] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16672–16683, May 2024.

[8] J. Bi, Z. Wang, H. Yuan, J. Qiao, J. Zhang, and M. Zhou, "Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for Complex Optimization Problems," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, London, United Kingdom, pp. 7966–7972.

[9] U. Azad, B. K. Behera, E. A. Ahmed, P. K. Panigrahi and A. Farouk, "Solving Vehicle Routing Problem Using Quantum Approximate Optimization Algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7564–7573, Jul. 2023.

[10] Y. Yu, X. Liu, Z. Liu and T. S. Durrani, "Joint Trajectory and Resource Optimization for RIS Assisted UAV Cognitive Radio," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 10, pp. 13643–13648, Oct. 2023.

[11] X. Wang, H. Yao, T. Mai, Z. Xiong, F. Wang and Y. Liu, "Joint Routing and Scheduling With Cyclic Queuing and Forwarding for Time-Sensitive Networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3793–3804, Mar. 2023.

[12] B. Yang, W. Wu, D. X. Yang, H. Wang and X. You, "Nonuniform-Array-Based Integrated MIMO Communication and Positioning in Wireless Local Area Networks," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 4937–4951, Mar. 2023.

[13] Y. Li, D. Wang, T. Sun, X. Duan, and L. Lu, "Solutions for Variant Manufacturing Factory Scenarios Based on 5G Edge Features," *2020 IEEE International Conference on Edge Computing (EDGE)*, 2020, pp. 54–58.

[14] M. Ghasemi, M. Zare, P. Trojovský, R. V. Rao, E. Trojovská, and V. Kandasamy, "Optimization based on the smart behavior of plants with its engineering applications: Ivy algorithm," *Know.-Based Syst.*, vol. 295, no. C, pp. 1–36, Jul. 2024.

[15] J. Li, R. Liu and R. Wang, "Elastic Strategy-Based Adaptive Genetic Algorithm for Solving Dynamic Vehicle Routing Problem With Time Windows," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 13930–13947, Dec. 2023.

[16] P. J. Barnawal, V. N. Lal and R. K. Singh, "A Dual Half-Active Bridge Resonant Converter for Solar PV Integration Using Grey Wolf Optimization," *IEEE Transactions on Industry Applications*, vol. 60, no. 5, pp. 7087–7097, Sept.-Oct. 2024.

[17] Y. Wang, F. Hu, H. Xu and J. Zeng, "A Multigroups Cooperative Particle Swarm Algorithm for Optimization of Multivehicle Path Planning in Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 11, no. 22, pp. 35839–35851, Nov. 2024.

[18] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for High-dimensional Problems," *Information Sciences*, vol. 630, no. 56, pp. 463–481, Jun. 2023.

[19] J. Zhai, J. Bi, H. Yuan, M. Wang, J. Zhang, Y. Wang, and M. Zhou, "Cost-Minimized Microservice Migration With Autoencoder-Assisted Evolution in Hybrid Cloud and Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 40951–40967, Dec. 2024.

[20] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm to Solve High-Dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 676–689, Aug. 2022.