# Multi-objective partial computation offloading for edge intelligence with heterogeneous components

Baoyu Xu[1,2], Yancheng Ruan[2], Tianyu Qi[3], Guobing Zou[2], Xiaoyang Kang[1,*], Lihua Zhang[1,*]
[1]College of Intelligent Robotics and Advanced Manufacturing, Fudan University, Shanghai, China
[2]School of Computer Engineering and Science, Shanghai University, Shanghai, China
[3]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
Email: byxu@shu.edu.cn, 18918694350@163.com, qitianyu@mail.ustc.edu.cn, gbzou@shu.edu.cn,
xiaoyang_kang@fudan.edu.cn, lihuazhang@fudan.edu.cn

2025 IEEE 31th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3315-4901-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICPADS67057.2025.11323179

***Abstract*—Edge intelligence, the fusion of edge computing and artificial intelligence (AI), drives the advancement of intelligent Internet of Things (IoT). Since AI applications are often data- and computation-intensive, resource-scarce edge devices need to migrate data to resource-rich edge servers through computation offloading to meet requirements such as energy efficiency and low latency. Existing studies often focus on CPU-based edge systems and neglect the impacts of other components, such as memory, on offloading. From a parallel processing perspective, this article establishes a system model and a multi-objective optimization model for edge intelligence systems with heterogeneous components, including diverse processors, memory, network, and applications, to minimize system energy consumption, total execution time, and the workload ratio of edge servers. A multi-objective optimization algorithm integrating archive initialization, hybrid perturbation, clustering, and modified simulated annealing is proposed and validated through experiments using real-world software and hardware. Results demonstrate that the proposed algorithm significantly outperforms comparative algorithms in terms of inverted generational distance, pure diversity, and runtime while revealing the influence of application characteristics on offloading performance.**

***Index Terms*—edge intelligence, computation offloading, multi-objective optimization.**

## I. Introduction

Edge computing is a special distributed computing paradigm, in which computation and communication resources from the cloud are pushed to the network edge to obtain faster response for end users [1]. Recently, edge computing is emerging in AI-driven IoT scenarios, including driverless, intelligent factories, and smart homes, which are called edge intelligence (EI). EI improves processing efficiency and enhances data privacy by combining AI and edge computing [2]. However, resource-scarce edge devices (EDs) in EI often lack sufficient resources for data-intensive or computation-intensive AI applications. To address this, computation offloading, transferring partial or full data from EDs to edge servers (ESs), provides an effective solution.

Due to the diversity of EDs and AI applications, EI systems are inherently heterogeneous [3]. Existing research on computation offloading primarily focuses on system models and offloading algorithms for homogeneous edge systems. For system modeling, studies [4]–[7] assume CPUs as sole processors and calculate energy consumption and latency based on required CPU cycles. However, EDs with GPU or ARM-based processors are ubiquitous, which results in the system models not aligning with real-world scenarios. In addition, other heterogeneous components (e.g., memory) in edge systems also contribute to energy and latency, yet their impact is often neglected. Consequently, offloading algorithms derived from these models tend to be inaccurate.

Offloading algorithms are categorized into single-objective and multi-objective optimization. The former optimizes a single objective(e.g., minimizing energy consumption), while the latter seeks optimal trade-offs among conflicting objectives. For instance, minimizing application latency, energy consumption of ED (EED), and resource utilization of ES (RES) simultaneously, computation offloading reduces EED but raises RES, while additional transmission latency may degrade application latency. To address this, some studies convert multi-objective optimizations into single-objective optimizations (e.g., via weighting methods) to identify optimal Pareto solutions. However, determining accurate weights for conflicting objectives remains challenging [8]. In addition, multi-objective algorithms effective in standard test cases may underperform in specific scenarios.

This study aims to construct a practical system model for the EI system and design a multi-objective optimization algorithm for partial computation offloading. There are some challenges involved. Heterogeneous components require a unified mathematical model that accounts for component interdependencies. For example, actual processor performance is affected by memory bandwidth and application complexity. Multi-objective offloading in edge systems is often a complex combinatorial problem, and some classic algorithms often fail in specific scenarios. Our main contributions are as follows.

- A system model for EI is proposed, characterizing heterogeneous EDs and applications. The model captures the dependencies among processors, memory, applications, and network via parallel processing and Roofline approach, profiles energy/latency at system and application levels, and establishes a multi-objective optimization model to minimize system energy consumption, total

* Corresponding author.

execution time, and the workload ratio of edge servers.

- A multi-objective improved simulated annealing (MOISA) is designed for offloading. MOISA extends simulated annealing from single-objective optimization to multi-objective optimization by integrating an external archive and an archive-based acceptance probability function, accelerates convergence via greedy rule-based archive initialization, balances global and local search through hybrid perturbation, and enhances search space via capacity-relaxed cluster based on elitism.
- Experiments with real-world deep learning models and hardware demonstrate that MOISA outperforms comparative algorithms in terms of inverted generational distance, pure diversity, and runtime. Furthermore, application-level energy consumption and latency analysis reveal the impact of application characteristics on offloading.

The rest of the article is organized as follows. Section II reviews related work. Section III describes the system model and problem formulation. Section IV proposes MOISA. Section V presents experimental results and analysis. Section VI concludes the article.

## II. Related work

### A. Heterogenous edge systems in offloading

Literatures [4]–[7] focus on edge computing, where EDs utilize CPUs as sole processors, and the energy and latency of applications on the EDs are calculated based on CPU cycles. Literatures [9], [10] use frames processed per unit time for processor performance, which establishes a unified mathematical model for heterogeneous processors, but neglects the costs of energy and latency in other components, such as memory.

Some studies focus on edge intelligence [6], [9]–[11]. Literature [6] employs multitask offloading to migrate deep learning networks to the edge or cloud. The edge devices are only equipped with Raspberry Pi. Literature [11] adopts a snapshot-based approach to offload the app of deep neural networks (DNNs) between the edge server and client. Several Convolutional NNs (CNNs) and a client board are used in experiments to validate the approach.

### B. Multi-objective optimization for offloading

Some algorithms have been employed for offloading optimization in edge computing. Genetic-inspired meta-heuristic algorithm minimizes the cost of task offloading and resource allocation for workflows in edge computing [12]. Grey wolf optimization optimizes energy and latency for compute-intensive applications [13]. Whale optimization algorithm addresses latency, energy consumption, the degree of privacy leakage, and their weighted sum [14]. The studies convert multi-objective optimization into single-objective optimization through the weighted sum method. However, weight selection remains challenging, especially in complex scenarios.

To identify optimal offloading, fast and elitist nondominated sorting genetic algorithm (NSGA-II) balances energy and latency in Internet of Things [15], and a modified NSGA-II optimizes latency, energy, and task completion quality

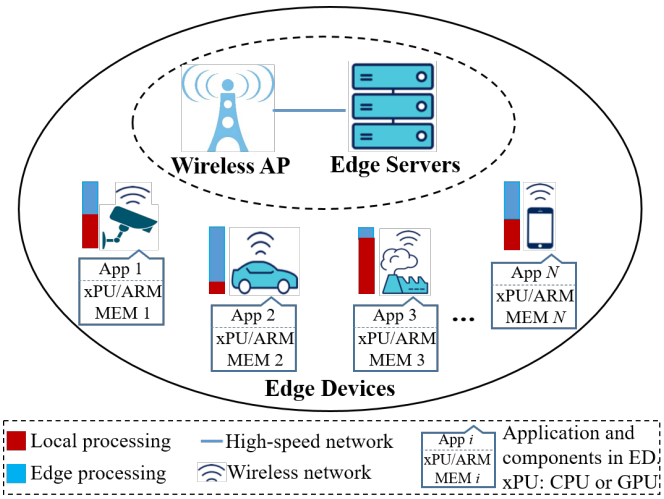



Fig. 1: System architecture of edge intelligence with heterogeneous components

in Internet of Vehicles [16]. A hybrid algorithm combining multi-objective particle swarm optimization (MOPSO) and NSGA-II minimizes latency, maximizes coverage, and reduces OPEX in edge systems [17]. Yadav et al. formulate edge computing offloading as a dual-objective optimization problem to minimize time and energy, using multi-objective grey wolf optimizer (MOGWO) for optimal decisions [18]. Jiang et al. enhance MOGWO for dependent task offloading, surpassing MOEA/D and MOPSO in convergence, diversity, and coverage [19]. These algorithms are scenario-specific and validated in homogeneous edge environments.

## III. System model and Problem formulation

This section models edge intelligence systems (EIS) and formulates computation offloading as a multi-objective optimization problem.

### A. System architecture

In our work, EIS is composed of $N$ EDs and $H$ ESs deployed at the edge. ESs process data offloaded from EDs through an Access Point(AP). Data is transmitted between EDs and ESs via wireless networks. Since ESs are often supplied by the same vendor and exhibit homogeneity similar to cloud servers [20], they are assumed as homogeneous. Figure 1 shows the system architecture.

Among EDs, there are not only differences in the peak performance of processors and memory but also variations in the energy consumption of a single operation on processors and memory. Therefore, the $i$ th ED is represented as $ED_i = \{c_i^{peak}, b_i^{peak}, e_i^f, e_i^m, P_i^{static}\}, i \in \{1..N\}$, where $c_i^{peak}$ and $b_i^{peak}$ denote the peak floating-point performance (flops/second) and peak memory bandwidth (bytes/second), respectively. $e_i^f$ and $e_i^m$ represent the energy consumption per floating-point operation (flop) and per memory operation (mop), respectively. $P_i^{static}$ is a fixed power regardless of whether an operation is performed or not. Let $c_{es}^{peak}$ and $b_{es}^{peak}$

denote the peak flop and peak memory bandwidth of ES, respectively.

### B. Application model

Applications in EIS are also different. The application running on the $i$ th ED is defined as $app_i = \{f_i, m_i\}, i \in \{1..N\}$, where $f_i$ and $m_i$ denote the number of flops and mops while executing a unit data of application $i$, such as a frame, respectively. $m_i$ is equal to the number of words transmitted. All applications are data-partitioned and submitted in EDs to leverage the parallelism of EDs and ESs.

### C. Latency model

The edge system adopts a wireless network which is the frequency-division duplex model. The transmission rate of ED $i$ is defined as $r_i$. According to [7], it is given by Equation 1.

$$r_i = \mu_i N_c W_c \log_2 \left(1 + P_i^t d_i^{-v} \mid \gamma_c \mid^2 / N_0\right) \tag{1}$$

Where $N_c$ denotes the total number of network channels, $W_c$ denotes the bandwidth of the channel. $d_i$ and $v$ represent the length and the loss exponent of the path between ED $i$ and its serving AP, respectively. $d_i^{-v}$ denotes the path loss. $\gamma_c$ denotes the channel fading coefficients. $N_0$ denotes white Gaussian noise power. $P_i^t$ is the transmit power of ED $i$. $\mu_i$ denotes the proportion of bandwidth in channels for ED $i$.

For ED $i$, the amount of downlink data is often ignored since it is much less than the amount of uplink data [21]. The amount of data uploaded is $t\beta_u\lambda_i s_i res_i$, where $\beta_u$ is the overhead of data transmission in an uplink channel and $res_i$ is the size of unit data. $s_i$ is the number of unit data captured per second by ED $i$. $\lambda_i$ denotes the ratio of the amount offloaded to its total amount, called offloaded ratio. Therefore, transmission latency is defined as $T_i^{tran}$, as follows.

$$T_i^{tran} = t\beta_u\lambda_i s_i res_i / r_i \tag{2}$$

Let $c_i^a$ denote the attainable performance of application $i$ executed on ED $i$. According to Roofline model [22], $c_i^a$ depends on $c_i^{peak}$ and $b_i^{peak}$, while being constrained by the characteristics of the application, including $f_i$ and $m_i$. $c_i^a$ is given by Equation 3.

$$c_i^a = \min\{c_i^{peak}, b_i^{peak} \times (f_i/m_i)\} \tag{3}$$

From the parallel perspective, flops, mops, and network transmission overlap maximally in the best case while applications are executing. The local latency of application $i$ is defined as $T_i^{local}$ and is caculated by Equation 4.

$$T_i^{local} = \max\{(1-\lambda_i)s_i t f_i / c_i^a, (1-\lambda_i)s_i t m_i / b_i^{peak}, T_i^{tran}\} \tag{4}$$

Where $(1-\lambda_i)s_i t f_i / c_i^a$ and $(1-\lambda_i)s_i t m_i / b_i^{peak}$ are times to execute the flops and mops of application $i$ on ED, respectively. Combing Equation 2, $T_i^{local}$ is obtained.

For ES, let $c_{i,es}^a$ denotes the attainable performance of the application $i$ in the edge server. According to the Roofline model, $c_{i,es}^a$ is given by Equation 5.

$$c_{i,es}^a = \min\{c_{es}^{peak}, b_{es}^{peak} \times (f_i/m_i)\} \tag{5}$$

Let $T_i^{es}$ denotes the latency of application $i$ in the best case when application $i$ is executed in ES. $T_i^{es}$ is calculated by Equation 6.

$$T_i^{es} = \max\{\lambda_i s_i t f_i / c_{i,es}^a, \lambda_i s_i t m_i / b_{es}^{peak}\} \tag{6}$$

Where $\lambda_i s_i t f_i / c_{i,es}^a$ and $\lambda_i s_i t m_i / b_{es}^{peak}$ denote times to execute flops and mops of application $i$ on ES, respectively.

Considering the latency of tranfering data from ED to ES and the parallel execution of the ED and ES, the latency of application $i$ in edge system is defined as $T_i$, as follows.

$$T_i = \max\{T_i^{local}, T_i^{es} + T_i^{tran}\} \tag{7}$$

The total latency of all applications is defined as $T$, referred to as total execution time, which is obtained via Equation 8.

$$T = \sum_{i=1}^{N} T_i \tag{8}$$

### D. Energy model

Due to limited ED battery life, we focus on energy consumed in ED, which mainly consists of data transmission, data processing (mainly from flops and mops), and static energy.

Let $e_i^{process}$ denotes the energy costed by data processing in ED $i$. For all EDs, the total energy of data processing in slot $t$ is defined as $E^{process}$. They are calculated as follows.

$$E^{process} = \sum_{i=1}^{N} e_i^{process} = \sum_{i=1}^{N} t s_i (1-\lambda_i)(e_i^f f_i + e_i^m m_i) \tag{9}$$

Where $ts_i(1-\lambda_i)$ denotes the number of unit data processed locally within time slot $t$. $(e_i^f f_i + e_i^m m_i)$ denotes the energy consumed by flops and mops per unit data on ED $i$.

Transmission energy for all EDs is shown as follows.

$$E^{tran} = \sum_{i=1}^{N} (P_i^0 + k_i^t N_c \mu_i P_i^t) T_i^{tran} \tag{10}$$

Where $P_i^0$ and $k_i^t$ denote idle power and amplifier coefficient for the network module, respectively. Combining Equation 2, the total energy of transmission can be obtained.

Suppose static energy is linear in $T_i$. $T_i P_i^{static}$ denotes the static energy of ED $i$. Total energy consumed in all EDs is called system energy consumption ($E$) that is calculated by Equation 11.

$$E = E^{process} + E^{tran} + \sum_{i=1}^{N} T_i P_i^{static} \tag{11}$$

### E. The workload ratio of edge servers (WRE)

$WRE$ is an indicator of ESs' current processing capacity. A lower $WRE$ indicates that ESs can handle more workloads. $WRE$ is calculated via Equation 12.

$$WRE = c^w / C_{max} = \sum_{i=1}^{N} f_i \min(\lambda_i s_i, r_i / res_i) / C_{max} \tag{12}$$

$c^w$ and $C_{max}$ respectively denote the number of flops required to process the all data offloaded to ESs within one second and the maximum number of flops that are processed by ESs within one second.

### F. Multi-objective optimization problem formulation

In EI, offloading data from EDs to ESs reduces $T$ and $E$ but increases $WRE$. Thus, minimizing $T$, $E$, and $WRE$ is conflicting. We formulate the multi-objective optimization problem in EI as $MEI$ that is defined as follows.

$$\begin{aligned} MEI: &\min_{\lambda,\mu}(T, E, WRE)^T \\ &\lambda = \{\lambda_1, ..., \lambda_i, ..., \lambda_N\} \\ &\mu = \{\mu_1, ..., \mu_i, ..., \mu_N\} \end{aligned} \tag{13}$$

***subject to***

$$C1: 0 \leq \lambda_i \leq 1, i = 1, 2, ..., N \tag{14}$$

$$C2: 0 \leq \mu_i \leq 1, i = 1, 2, ..., N \tag{15}$$

$$C3: \sum_{i=1}^{N} \mu_i \leq 1 \tag{16}$$

$$C4: \sum_{i=1}^{N} f_i \min(\lambda_i s_i, r_i/res_i) \leq C_{max} \tag{17}$$

In $MEI$, constraints $C1$ and $C2$ bound offloaded ratio and the proportion of bandwidth for each ED to $[0, 1]$. Constraint $C3$ ensures that the total ratio of bandwidth across all EDs does not exceed 1, to prevent surpassing the total network bandwidth. Constraint $C4$ guarantees that the total flops required by offloaded data do not exceed $C_{max}$.

According to Equations 7, 9, 10, and 11, $E$ is nonlinear with respect to $\lambda$ and $\mu$, and often discontinuous, especially at the boundary of the domain. Therefore, minimizing $E$ is a nonlinear programming (NLP) problem that is NP-hard in nature [23], [24]. Furthermore, $T$ and $WRE$ are correlated with $E$, and minimizing them simultaneously along with $E$ leads to conflicts. Consequently, $MEI$ is more complex than minimizing $E$ alone. To address this, the next section proposes a multi-objective optimization algorithm based on an improved simulated annealing (SA) to achieve computation offloading.

## IV. Optimal solution

### A. Constraint elimination

A penalty function method [25] is first employed to eliminate the constraints in $MEI$. The penalty $G(x)$ for constraint violations is:

$$G(x) = \sum_{i=1}^{Con_{\neq}} [\max\{0, -g_i(x)\}]^{\gamma_1} + \sum_{j=1}^{Con_{=}} \mid h_j(x) \mid^{\gamma_2} \tag{18}$$

Where solution $x$ comprises decision variables $\mu$ and $\lambda$. $g_i(x)$ and $h_j(x)$ represent inequality and equality constraints, with $\gamma_1, \gamma_2 \in \{1, 2\}$. $Con_{\neq}$ and $Con_{=}$ denote the number of inequality and equality constraints. $T$, $E$, and $WRE$ are transformed to unconstrained objectives $T^{ce}$, $E^{ce}$, and $WRE^{ce}$ by Equation 19.

$$(T^{ce}, E^{ce}, WRE^{ce})^T = (T, E, WRE)^T + \varphi G(x) \tag{19}$$

where $\varphi$ is a large penalty parameter. This yields the unconstrained problem:

$$MEI - CE: \min_{\lambda,\mu}(T^{ce}, E^{ce}, WRE^{ce})^T \tag{20}$$

**Algorithm 1** MOISA

**Input:**
$ED_i, app_i, s_i, i \in [1, N]$; $maxNum$;
$T_s$ : starting temperature; $T_f$ : final temperature;
$maxIter$ : the number of iterations per temperature;
$sf$: switching factor; $cr$ : cooling rate

**Output:**
$archive$

Initialize archive using AIBG
$temp \leftarrow T_s$
**while** $temp \geq T_f$ **do**
  **for** $iter = 1$ to $maxIter$ **do**
    **if** $iter/maxIter \leq sf$ **then**
      obtain $x^{new}$ with SBX
    **else**
      obtain $x^{new}$ with DE
    **end if**
    $A \leftarrow$ solutions in archive dominated by $x^{new}$
    $B \leftarrow$ solutions in archive dominate $x^{new}$
    $aprob$ is obtained by (21)
    **if** $x^1$ dominates $x^{new}$ **then**
      **if** $rand(0, 1) < aprob$ **then**
        put $x^{new}$ in archive
        **if** $archive.size \geq maxNum$ **then**
          CBE
        **end if**
      **end if**
    **else**
      **if** $A.size > B.size$ **then**
        put $x^{new}$ in archive
        **if** $archive.size \geq maxNum$ **then**
          CBE
        **end if**
      **else**
        **if** $rand(0, 1) < aprob$ **then**
          put $x^{new}$ in archive
          **if** $archive.size \geq maxNum$ **then**
            CBE
          **end if**
        **end if**
      **end if**
    **end if**
  **end for**
  $temp \leftarrow cr \times temp$
**end while**
return $archive$

### B. Multi-objective improved simulated annealing (MOISA)

SA is a heuristic algorithm inspired by physical annealing processes, using the Metropolis criterion to probabilistically escape local optima and converge to global optima [26].

SA is inherently a single-objective algorithm and can be extended to multi-objective optimization by incorporating an external archive that stores non-dominated solutions [27].

Meanwhile, unlike the acceptance probability function based on the difference of single-objective values [26], an archive-based acceptance probability function ($aprob$) [28] for a new solution ($x^{new}$) is introduced as follows.

$$aprob = \exp((A.size - B.size)/(archive.size \times temp)) \tag{21}$$

Where $archive.size$ refers to the number of solutions in the archive, $temp$ represents the current temperature, $A$ is the set of archive solutions dominated by $x^{new}$, and $B$ is the set of archive solutions dominating $x^{new}$. $aprob$ reflects the dominance relationships between $x^{new}$ and other solutions in the archive.

Algorithm 1 presents Multi-Objective Improved Simulated Annealing (MOISA) based on the archive and $aprob$. In the algorithm, line 1 initializes the archive. Lines 2-4 determine the number of iterations. In each iteration, lines 5–9 generate $x^{new}$ from a current solution ($x^1$) selected from the archive via hybrid perturbation. Lines 13–19 describe operations when $x^{new}$ is dominated by $x^1$, while lines 21–33 define operations when $x^{new}$ dominates $x^1$. If $x^{new}$ is dominated or $A.size \leq B.size$, its acceptance is determined by the Metropolis criterion incorporating $aprob$. Upon accepting a new solution, lines 16-18, 23-25, and 29-31 constrain the archive capacity while enhancing diversity through a modified cluster. The implementation details of archive initialization, hybrid perturbation, and cluster are subsequently introduced.

*1) Archive initiation based on a greedy rule (AIBG):* To accelerate convergence, AIBG generates feasible initial solutions for the archive. For each solution, to satisfy constraints $C2$ and $C3$, AIBG first assigns $N$ random values between 0 and 1 to the proportion of bandwidth of EDs ($\mu$) and ensures that the sum of the values does not exceed 1. Since offloading transfers data to ESs to reduce energy and latency consumed by EDs, larger random values are assigned to EDs with greater consumption, which increases their opportunities to offload more data and consequently reduce their energy consumption and latency. To enhance diversity, only a subset of the initial archive undergoes the initialization mentioned above. The remainder is generated randomly under the constraints in $MEI$.

*2) Solution generation by hybrid perturbation (SHP):* Our work employs Simulated Binary Crossover (SBX) and Differential Evolution (DE) perturbations for generating new solutions. In SBX, two solutions are randomly selected from the archive, denoted as $x^1$ and $x^2$, where $x^1$ is the current solution. $x^{new}$ is obtained through Equation 22.

$$x^{new} = 0.5 \times [(1+\beta) \times x^1 + (1-\beta) \times x^2] \tag{22}$$

Where $\beta$ is dynamically obtained from [29].

In DE, three solutions $x^1$, $x^2$, and $x^3$ are randomly selected from the archive, where $x^1$ serves as the current solution. $x^{new}$ is generated by Equation 23.

$$x^{new} = x^1 + F \times (x^2 - x^3) \tag{23}$$

Here, $F$ represents the scaling factor. Subsequently, random gene position $k$ in $x^{new}$ is replaced with the corresponding value from $x^1$ to preserve partial genetic information, which is called the crossover operation and is shown in Figure 2.

A switching factor ($sf$) coordinates SBX and DE. Large-magnitude perturbation (SBX) is applied in early iterations to enhance diversity, while small-magnitude perturbation (DE) is utilized in later iterations to facilitate convergence.

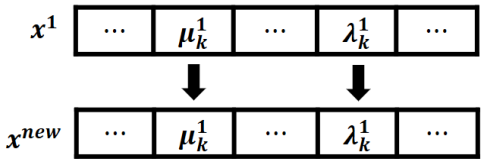



Fig. 2: Crossover operation in DE.

*3) Capacity-relaxed cluster based on elitism (CBE):* During iterations, solutions in the archive tend to increase. Most algorithms maintain the archive capacity at the initial capacity ($AIC$) during optimal solution clustering. Based on the Metropolis criterion, however, the archive capacity should be relaxed to retain more suboptimal solutions to expand the search space and enhance solution diversity. Our work sets a threshold $maxNum$ (exceeding $AIC$) and employs the elitism strategy from NSGA-II [27] for archive clustering and updating. When exceeding $maxNum$, the archive is updated by retaining all non-dominated solutions ($\mathcal{F}$). If $|\mathcal{F}| < maxNum$, it is supplemented by subsequent Pareto fronts; otherwise, keeping the top $maxNum$ solutions from $\mathcal{F}$, sorted by crowding distance. Crowding distance sums each solution's elementary distances across objectives [27].

## V. Evaluation

### A. The setting of system, application, and algorithm

TABLE I: Parameters of EDs and ES. Processor peak (P. pk.) and memory bandwidth (M. bw.) is vendor's claimed peak.

| EDs/ES | P. pk. (Gflop/s) | M. bw. (GB/s) | $e^f$ (pJ/flop) | $e^m$ (pJ/mop) | $P^{static}$ (w) |
|---|---|---|---|---|---|
| Arndale (A15) | 27.2 | 12.8 | 107 | 386 | 5.5 |
| Arndale (T-604) | 72 | 12.8 | 84.2 | 518 | 1.28 |
| NUC (I3) | 57.6 | 25.6 | 14.7 | 418 | 16.5 |
| **TX2 (A57)** | 96 | 59.7 | 36.6 | 612.3 | 1.9 |
| NVIDIA (GF100) | 1580 | 192 | - | - | - |

TABLE II: Flops and mops of DL models

| Models | Input dimension | Flops (Mflops) | Mops (Mbytes) |
|---|---|---|---|
| AlexNet | 224 × 224 | 1005.89 | 181.18 |
| SqueezeNet | 224 × 224 | 818.92 | 23.44 |
| MobileNet | 224 × 224 | 567.73 | 31.43 |
| MobileNet v2 | 224 × 224 | 300.77 | 38.72 |
| MobileNet v3 Large | 224 × 224 | 220.30 | 33.70 |
| MobileNet v3 Small | 224 × 224 | 59.24 | 12.58 |
| VGG16 | 224 × 224 | 15466.21 | 569.92 |
| VGG19 | 224 × 224 | 19628.01 | 595.05 |
| GoogLeNet | 224 × 224 | 1581.66 | 40.52 |
| Inception v2 | 224 × 224 | 1719.33 | 51.22 |

In experiments, EDs comprise devices from Arndale, NUC, and NVIDIA TX2, which utilize ARM Cortex-A15 (A15), Mali T-604 (T-604), Intel Core i3-3217U (I3), and ARM Cortex-A57 (A57), respectively. ES utilizes NVIDIA GF100. TX2 parameters are measured experimentally; others are obtained from [30]. Table I lists them.

Deep learning (DL), a powerful subset of machine learning algorithms, has become a key technology advancing artificial intelligence through automated feature extraction and hierarchical data representation. Therefore, DL models are adopted in experiments. Table II lists flops and mops for ten DL models [31] processing a frame with 224×224 resolution. According to [7], network parameters are as follows: $W_c = 10MHz$, $N_c = 15$, $\gamma_c = 0.98$, $N_0 = 1.6 \times 10^{-11}$, $v = 4$, $\beta_u = 1$, $P_i^t = 0.1W$, $P_i^0 = 0.4W$, $k_i^t = 18$. Time slot=1s. In MOISA, $sf = 0.7$, $F = 0.4$, $T_s = 100\,^{\circ}\mathrm{C}$, $T_f = 0.1\,^{\circ}\mathrm{C}$, $maxIter = 10$, and $maxNum = 2AIC$. Experiments are conducted on ThinkBook K3 equipped with an Intel I7 CPU ($2.8GHz$) and $16GB$ RAM.

TABLE III: Metrics for five algorithms as N increases.

| N | Methods | IGD | HV | PD | Time(s) |
|---|---|---|---|---|---|
| 10 | MOISA | **0.0295** | 7.521 | **0.5566** | **24.457** |
| | NSGA-II | 0.7695 | 2.8613 | 0.0002 | 29.717 |
| | MOPSO | 0.0599 | 10.7602 | 0.3952 | 51.756 |
| | NSRS | 0.1792 | 3.074 | 0.3288 | 25.952 |
| | MOGWO | 0.6366 | 0.181 | 0.0029 | 25.599 |
| 20 | MOISA | **0.0807** | 19.3069 | **0.319** | **35.382** |
| | NSGA-II | 0.4249 | 38.4443 | 0.0015 | 64.647 |
| | MOPSO | 0.1129 | 14.2276 | 0.2197 | 116.496 |
| | NSRS | 0.1236 | 7.852 | 0.1467 | 57.811 |
| | MOGWO | 0.2338 | 30.0188 | 0.0773 | 57.31 |
| 30 | MOISA | **0.1814** | **36.6966** | **0.2496** | **37.861** |
| | NSGA-II | 0.5236 | 32.0126 | 0.0023 | 76.759 |
| | MOPSO | 0.2341 | 9.9557 | 0.1555 | 145.25 |
| | NSRS | 0.2241 | 6.7051 | 0.1511 | 72.258 |
| | MOGWO | 0.5458 | 0.9815 | 0.0124 | 78.057 |
| 40 | MOISA | **0.1601** | 25.1591 | **0.2837** | **48.488** |
| | NSGA-II | 0.5797 | 30.2472 | 0.0018 | 124.985 |
| | MOPSO | 0.3242 | 15.4261 | 0.109 | 231.102 |
| | NSRS | 0.3775 | 3.9982 | 0.0652 | 114.403 |
| | MOGWO | 0.573 | 0.6698 | 0.0085 | 118.801 |
| 50 | MOISA | **0.119** | 21.7372 | **0.3266** | **46.883** |
| | NSGA-II | 0.5421 | 23.6697 | 0.0032 | 132.499 |
| | MOPSO | 0.2695 | 10.0454 | 0.1375 | 273.095 |
| | NSRS | 0.315 | 4.1428 | 0.0859 | 136.295 |
| | MOGWO | 0.4767 | 3.4276 | 0.0434 | 133.875 |
| 60 | MOISA | **0.3123** | **24.4448** | **0.1058** | **53.474** |
| | NSGA-II | 0.5602 | 5.6843 | 0.0029 | 151.648 |
| | MOPSO | 0.3231 | 11.3816 | 0.0859 | 322.746 |
| | NSRS | 0.3358 | 4.5664 | 0.0845 | 156.586 |
| | MOGWO | 0.5819 | 0.2347 | 0.003 | 160.303 |
| 70 | MOISA | **0.1595** | **30.9386** | **0.2242** | **56.288** |
| | NSGA-II | 0.5229 | 30.554 | 0.0078 | 169.868 |
| | MOPSO | 0.3197 | 12.6445 | 0.089 | 349.355 |
| | NSRS | 0.3364 | 2.2859 | 0.0613 | 183.628 |
| | MOGWO | 0.4886 | 3.185 | 0.0403 | 172.912 |

### *B. Comparative algorithms and metrics*

Comparative algorithms include NSGA-II [27], MOPSO [19], and MOGWO [18], and NSRS modified modified Random Search [32] with nondominated sorting and external archive. The archive capacity is 40. The number of iterations is 500. All algorithms employ the same initialization, and then MOISA employs AIBG.

Evaluation Metrics include: Inverted Generational Distance (**IGD**) measures the average Euclidean distance from each point in the true Pareto front (PF) to the nearest solution in the algorithm-generated PF (APF) [8]. Lower IGD indicates better convergence/distribution. Hypervolume (**HV**) quantifies the volume of the objective space dominated by APF. Higher HV suggests better convergence and diversity. Pure Diversity (**PD**) evaluates diversity based on objective value differences, where higher PD indicates reduced clustering [33]. **Time** records algorithm runtime. Due to significant magnitude differences among the three optimization objectives, normalization is required before calculating IGD, HV, and PD.

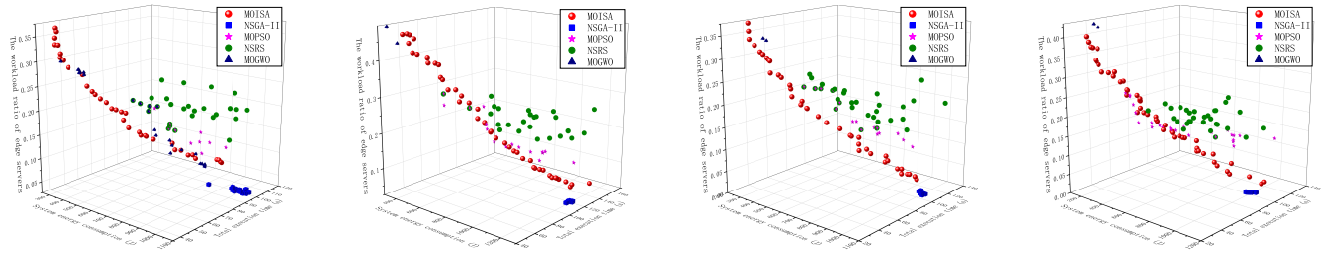
TABLE IV: Metrics for five algorithms as NetDis increases.

| NetDis(m) | Methods | IGD | HV | PD | Time(s) |
|---|---|---|---|---|---|
| 40 | MOISA | **0.1749** | **35.1116** | **0.2212** | **37.077** |
| | NSGA-II | 0.5525 | 35.7768 | 0.0023 | 95.351 |
| | MOPSO | 0.2619 | 13.7175 | 0.1497 | 195.270 |
| | NSRS | 0.2825 | 4.5596 | 0.1088 | 90.755 |
| | MOGWO | 0.5612 | 0.0274 | 0.0003 | 100.350 |
| 60 | MOISA | **0.182** | 28.3184 | **0.2408** | **40.245** |
| | NSGA-II | 0.576 | 38.3111 | 0.0011 | 98.293 |
| | MOPSO | 0.2021 | 13.7945 | 0.2408 | 202.970 |
| | NSRS | 0.304 | 4.0993 | 0.0924 | 95.921 |
| | MOGWO | 0.5012 | 4.1228 | 0.0379 | 102.331 |
| 80 | MOISA | **0.1383** | 18.9768 | **0.3195** | **40.416** |
| | NSGA-II | 0.5815 | 28.8778 | 0.0066 | 93.524 |
| | MOPSO | 0.2888 | 8.6876 | 0.146 | 188.646 |
| | NSRS | 0.298 | 3.9681 | 0.1036 | 86.757 |
| | MOGWO | 0.4806 | 4.955 | 0.0064 | 99.473 |
| 100 | MOISA | **0.1601** | 25.1591 | **0.2837** | **48.488** |
| | NSGA-II | 0.5797 | 30.2472 | 0.0018 | 124.985 |
| | MOPSO | 0.3242 | 15.4261 | 0.109 | 231.102 |
| | NSRS | 0.3775 | 3.9982 | 0.0652 | 114.403 |
| | MOGWO | 0.573 | 0.6698 | 0.0085 | 118.801 |
| 120 | MOISA | **0.1021** | **32.3557** | **0.2922** | **50.482** |
| | NSGA-II | 0.53 | 18.0073 | 0.0023 | 118.309 |
| | MOPSO | 0.1433 | 8.0088 | 0.2468 | 223.442 |
| | NSRS | 0.2858 | 4.0151 | 0.0957 | 113.615 |
| | MOGWO | 0.4039 | 4.9664 | 0.0083 | 110.574 |
| 140 | MOISA | **0.1757** | **36.4243** | **0.2932** | **43.657** |
| | NSGA-II | 0.5757 | 12.461 | 0.0006 | 103.890 |
| | MOPSO | 0.2746 | 8.9457 | 0.1743 | 200.855 |
| | NSRS | 0.3438 | 5.2842 | 0.0944 | 92.746 |
| | MOGWO | 0.3831 | 0.5558 | 0.0114 | 109.636 |
| 160 | MOISA | **0.1195** | **31.5951** | **0.2616** | **42.429** |
| | NSGA-II | 0.5215 | 19.7291 | 0.0244 | 104.185 |
| | MOPSO | 0.2395 | 9.5659 | 0.1564 | 201.005 |
| | NSRS | 0.2762 | 1.9582 | 0.0856 | 96.277 |
| | MOGWO | 0.4581 | 2.9851 | 0.0378 | 111.038 |

### *C. Performance comparison for algorithms*

Table III presents the metrics as the number of EDs (N) increases from 10 to 70, with frames per second (FPS) fixed at 30 and network distance (NetDis) between EDs and wireless AP at 100 meters. The table shows that MOISA achieves the minimum IGD, maximum PD, and shortest Time. For HV, MOISA outperforms other algorithms when N is 30, 60, and 70, but exhibits slightly inferior performance in other cases.

Table IV presents the variations in the metrics for the five algorithms as NetDis increases from 40 to 160 meters, with N fixed at 40 and FPS at 30. Consistent with the results in




(a) N=40 (b) N=50 (c) NetDis=80(m) (d) NetDis=100(m)

Fig. 3: Three-dimensional plots of the Pareto fronts for five algorithms with poor HV in MOISA.

Table III, MOISA demonstrates superior performance in IGD, PD, and Time compared to other algorithms. For HV, MOISA exhibits better performance at longer network distances.

The HV metric exhibits inconsistent performance, necessitating further investigation. Figure 3 shows partial three-dimensional plots of the Pareto fronts for five algorithms with poor HV in MOISA, as specified in Tables III and IV. The plots reveal that the PF points obtained by NSGA-II are clustered in a localized region, which demonstrates poor distribution. In addition, the PF points obtained by MOGWO are few and relatively clustered, which leads to inferior IGD and PD. Most of the PF points obtained by NSRS deviate from the minimum value of the objective, resulting in poor IGD. MOPSO can obtain PF points that are close to the minimum value of the objective. However, due to fewer points and uneven distribution in the objective space, its performance is inferior to that of MOISA. These figures show the advantages of MOISA in convergence, diversity, and distribution.

### *D. Application-level evaluations and analysis*

The method proposed in this paper captures not only the energy consumption and latency of systems (e.g., $E$ and $T$) but also those of applications, which facilitates the comprehensive understanding of application characteristics influencing energy consumption and latency.

Figure 4 illustrates the application energy consumption (AE) at the point with minimum $E$ on the PF of MOISA when N is 10. "No-off", "Off", and "ROD" represent the energy consumption/latency without computation offloading, with computation offloading, and their reduction ratio, respectively. The figure reveals that for high-resource models (e.g., VGG16 and VGG19), characterized by more flops and mops, offloading reduces application energy consumption by over 90%. In contrast, for low-resource models (e.g., MobileNet series), characterized by fewer flops and mops, offloading has negligible effects. Notably, the application energy of MobileNet v3 Small model remains constant post-offloading. This phenomenon occurs because offloading data from high-resource models with higher energy consumption can more effectively reduce the energy consumption of EDs. Figure 5 reveals that changes in application latency follow similar patterns to those of AE.

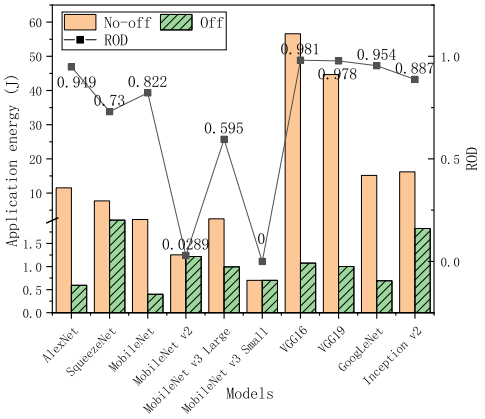



Fig. 4: Application energy consumption at the point with minimum $E$ on the MOISA's Pareto front.

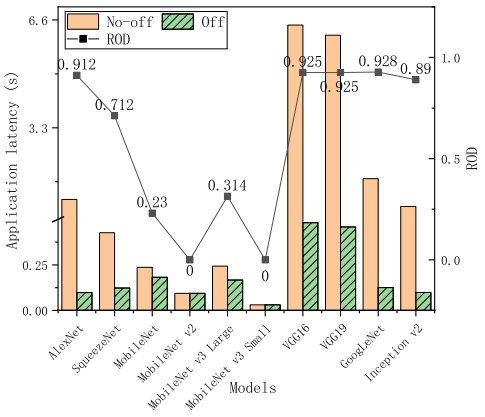



Fig. 5: Application latency at the point with minimum $T$ on the MOISA's Pareto front.

## VI. Conclusion

For edge intelligence, this article constructs a system model that analyzes the impact of heterogeneous components on computation offloading, presents a multi-objective optimization model minimizing system energy consumption, total execution time, and WRE, and proposes MOISA to achieve partial computation offloading. MOISA accelerates convergence by AIBG, balances global and local search capabilities with SHP,

and enhances solution diversity through capacity-relaxed CBE. The metrics of IGD, PD, and Time, along with 3D Pareto front plots, demonstrate that MOISA significantly outperforms the comparative algorithms, including NSGA-II, MOPSO, and MOGWO, in terms of convergence, diversity, distribution, and runtime. Analysis of application-level energy consumption and latency reveals that resource-intensive DL models benefit most from offloading. Future work will explore edge caching to address network bottlenecks.

## ACKNOWLEDGMENT

This work was supported by the Shanghai Municipal Science and Technology Major Project (grant No. 2021SHZDZX0103).

## REFERENCES

[1] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, pp. 7457–7469, 2020.

[2] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Empowering intelligence to the edge of network," *Proceedings of the IEEE*, vol. 109, pp. 1778–1837, 11 2021.

[3] P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, and K. Li, "A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches," *Computer Science Review*, vol. 53, 2024.

[4] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An a3c-based approach," *IEEE Transactions on Network Science and Engineering*, vol. 10, pp. 1326–1338, 2023.

[5] Z. Xue, C. Liu, C. Liao, G. Han, and Z. Sheng, "Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 72, pp. 6709–6722, 2023.

[6] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," *2017 2nd ACM/IEEE Symposium on Edge Computing, SEC 2017*, 2017.

[7] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, pp. 3774–3785, 2021.

[8] X. Zhu and M. C. Zhou, "Multiobjective optimized cloudlet deployment and task offloading for mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, pp. 15 582–15 595, 2021.

[9] Y. Huang, Y. Lu, F. Wang, X. Fan, J. Liu, and V. C. Leung, "An edge computing framework for real-time monitoring in smart grid," *Proceedings - 2018 IEEE International Conference on Industrial Internet, ICII 2018*, pp. 99–108, 2018.

[10] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," *Proceedings - IEEE INFOCOM*, vol. 2020-July, pp. 257–266, 2020.

[11] H. J. Jeong, I. Jeong, H. J. Lee, and S. M. Moon, "Computation offloading for machine learning web apps in the edge server environment," *Proceedings - International Conference on Distributed Computing Systems*, vol. 2018-July, pp. 1492–1499, 2018.

[12] L. Ma, Y. Zhang, J. Zhou, and G. Zhang, "A gene-inspired metaheuristic for scheduling workflow tasks in mobile edge computing-supported cyber–physical systems," *Journal of Systems Architecture*, vol. 151, p. 103136, 2024.

[13] A. Abbas, A. Raza, F. Aadil, and M. Maqsood, "Meta-heuristic-based offloading task optimization in mobile edge computing," *International Journal of Distributed Sensor Networks*, vol. 17, 2021.

[14] Z. Liu, J. Wang, Z. Gao, and J. Wei, "Privacy-preserving edge computing offloading scheme based on whale optimization algorithm," *Journal of Supercomputing*, vol. 79, pp. 3005–3023, 2023.

[15] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, pp. 4791–4803, 6 2019.

[16] Z. Sifeng, D. Haowei, Y. Yaxing, C. Hao, and Z. Hai, "Improved nsga-ii algorithm-based task offloading decision in the internet of vehicles edge computing scenario," *Multimedia Systems*, vol. 31, 2 2025.

[17] B. Bahrami, M. R. Khayyambashi, and S. Mirjalili, "Multiobjective placement of edge servers in mec environment using a hybrid algorithm based on nsga-ii and mopso," *IEEE Internet of Things Journal*, vol. 11, pp. 29 819–29 837, 2024.

[18] J. Yadav and Suman, "E-mogwo algorithm for computation offloading in fog computing," *Intelligent Automation and Soft Computing*, vol. 36, pp. 1063–1078, 2023.

[19] K. Jiang, H. Ni, R. Han, and X. Wang, "An improved multi-objective grey wolf optimizer for dependent task scheduling in edge computing," *International Journal of Innovative Computing, Information and Control*, vol. 15, pp. 2289–2304, 12 2019.

[20] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," *Proceedings - 2013 IEEE 7th International Symposium on Service-Oriented System Engineering, SOSE 2013*, pp. 494–502, 2013.

[21] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*. IEEE, 2016, pp. 1–6.

[22] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, pp. 65–76, 2009.

[23] B. Xu, Y. Ruan, C. Qiu, S. He, F. Shu, X. Kang, and L. Zhang, "Multilevel and energy-efficient partial computation offloading in heterogeneous edge intelligence," *IEEE Internet of Things Journal*, vol. 12, pp. 14 993–15 007, 2025.

[24] A. Daghayeghi and M. Nickray, "Decentralized computation offloading via multi-agent deep reinforcement learning for noma-assisted mobile edge computing with energy harvesting devices," *Journal of Systems Architecture*, vol. 151, p. 103139, 2024.

[25] A. Panda and S. Pani, "A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems," *Applied Soft Computing Journal*, vol. 46, pp. 344–360, 2016.

[26] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: A review and a new scheme," in *IEEE Workshop on Statistical Signal Processing Proceedings*, vol. 2021-July. IEEE Computer Society, 7 2021, pp. 101–105.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," Tech. Rep., 2002.

[28] R. Sengupta and S. Saha, "Reference point based archived many objective simulated annealing," *Information Sciences*, vol. 467, pp. 725–749, 10 2018.

[29] Q. Xiong, L. Dong, H. Chen, X. Zhu, X. Zhao, and X. Gao, "Enhanced nsga-ii algorithm based on novel hybrid crossover operator to optimise water supply and ecology of fenhe reservoir operation," *Scientific Reports*, vol. 14, 12 2024.

[30] J. Choi, M. Dukhan, X. Liu, and R. Vuduc, "Algorithmic time, energy, and power on candidate hpc compute building blocks," *Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS*, pp. 447–457, 2014.

[31] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, 12 2021.

[32] L. Deng and S. Liu, "Deficiencies of the whale optimization algorithm and its validation method," *Expert Systems with Applications*, vol. 237, p. 121544, 2024.

[33] Y. Xiang, J. Guo, C. Jiang, H. Ma, and M. Liu, "Multi-objective five-element cycle optimization algorithm based on multi-strategy fusion for the bi-objective traveling thief problem," *Applied Sciences (Switzerland)*, vol. 14, 2024.