



Hybrid metaheuristics for selective inference task offloading under time and energy constraints for real-time IoT sensing systems

Abdelkarim Ben Sada¹ · Amar Khelloufi¹ · Abdenacer Naouri¹ · Huansheng Ning¹ · Sahraoui Dhelim²

Received: 22 February 2024 / Revised: 13 May 2024 / Accepted: 17 May 2024 / Published online: 19 June 2024
© The Author(s) 2024

Abstract

The recent widespread of AI-powered real-time applications necessitates the use of edge computing for inference task offloading. Power constrained edge devices are required to balance between processing inference tasks locally or offload to edge servers. This decision is determined according to the time constraint demanded by the real-time nature of applications, and the energy constraint dictated by the device's power budget. This problem is further exacerbated in the case of systems leveraging multiple local inference models varying in size and accuracy. In this work, we tackle the problem of assigning inference models to inference tasks either using local inference models or by offloading to edge servers under time and energy constraints while maximizing the overall accuracy of the system. This problem is shown to be strongly NP-hard and therefore, we propose a hybrid genetic algorithm (HGSTO) to solve this problem. We leverage the speed of simulated annealing (SA) with the accuracy of genetic algorithms (GA) to develop a hybrid, fast and accurate algorithm compared with classic GA, SA and Particle Swarm Optimization (PSO). Experiment results show that HGSTO achieved on-par or higher accuracy than GA while resulting in significantly lower scheduling times compared to other schemes.

Keywords Selective sensing · Edge computing · Machine learning · Task offloading · Genetic algorithms

1 Introduction

In recent years, the proliferation of Internet of Things (IoT) devices and the exponential growth of data generated at the edge of networks have fueled the emergence of edge computing as a promising paradigm for efficient data processing and analysis. Edge computing leverages the computational resources available at the network periphery, closer to where data is generated, to provide low-latency and high-throughput services for various applications, ranging from real-time analytics to augmented reality [1, 2].

Central to the efficacy of edge computing is the efficient allocation of computational tasks among edge devices and

centralized servers, aiming to strike a balance between resource utilization, latency reduction, and energy efficiency. Particularly, in inference tasks, where predictive models are deployed to analyze data and make real-time decisions, the allocation of tasks to appropriate computational resources becomes crucial [3].

The landscape of artificial intelligence (AI) has witnessed remarkable advancements, driven by breakthroughs in AI training algorithms and the availability of powerful hardware accelerators. As a result, the deployment of multiple compact AI inference models at the edge has become increasingly prevalent. These models, characterized by their small size and efficient computational footprint, offer significant advantages in terms of resource utilization, latency reduction, and energy efficiency. By leveraging techniques such as model distillation, quantization, and pruning, researchers have been able to create highly optimized inference models that are well-suited for deployment on resource-constrained edge devices [4]. Furthermore, the proliferation of specialized hardware accelerators, such as GPUs, TPUs, and FPGAs, has further accelerated this trend, enabling edge devices to perform

✉ Sahraoui Dhelim
sahraoui.dhelim@ucd.ie

¹ School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China

² School of Computer Science, University College Dublin, Dublin, Ireland

complex AI inference tasks with unprecedented speed and efficiency. As such, the availability of multiple compact AI inference models at the edge presents a unique opportunity to enhance the performance and scalability of edge computing systems while catering to the diverse needs of emerging edge applications [5–8].

In this paper, we address the challenging problem of inference task scheduling and offloading in edge computing environments characterized by time and energy constraints. Specifically, we consider scenarios where edge devices are equipped with multiple small-sized inference models, each varying in accuracy and inference times (see Fig. 1). The objective is to assign inference tasks to either local edge models or centralized edge server models in a manner that maximizes accuracy while adhering to stringent time and energy constraints imposed by the application domain.

The problem of inference task offloading under time and energy constraints in edge computing is inherently complex and challenging, with its computational complexity exacerbated by the NP-hard nature of the problem. As a combinatorial optimization problem, determining the optimal allocation of inference models for inference tasks between local models and edge server models involves exploring a vast search space of possible task assignments while simultaneously considering multiple conflicting objectives, such as maximizing accuracy, while respecting time and energy constraints. The NP-hardness of the problem implies that finding an exact solution within a reasonable amount of time becomes increasingly difficult as the size of the problem instance grows, rendering traditional exact algorithms impractical for real-world deployment [9].

The significance of solving this problem lies in its direct impact on the performance, scalability, and cost-effectiveness of edge computing systems across various application domains. For instance, in healthcare applications, such as remote patient monitoring and real-time diagnosis, timely and accurate inference results are critical for

ensuring patient safety and well-being. Similarly, in autonomous vehicles and smart transportation systems, efficient inference task offloading can enhance decision-making processes, leading to improved traffic management, accident prevention, and passenger safety. Furthermore, in industrial IoT (IIoT) applications, such as predictive maintenance and quality control, the ability to offload inference tasks optimally can result in substantial cost savings, productivity gains, and operational efficiency improvements [10, 11].

In literature, we find a scarcity of research concerning inference task scheduling and offloading under constraints mainly due to the novelty of the problem at hand. The work by [12] focuses on data selection based on the likelihood of low inference accuracy to offload to an edge server under an energy constraint. This work points in the direction of the more general problem this paper is tackling. However, they only take advantage of a single inference model along with a single edge server which is a smaller special case of the bigger problem. Additionally, the time dimension which is important for real-time applications, is not addressed. The work in [13] provides a broader coverage of the problem at hand by leveraging the dynamicity offered by deploying multiple inference models in the edge device while respecting a time constraint. Their work however, lacks the dimension of energy, which is important for battery-powered edge devices. Furthermore, their system only handles offloading to a single edge server. This leaves a research gap, in which the broader problem needs to be addressed where both time and energy dimensions are respected while maximizing the accuracy of inference by assigning inference models from a set of local and edge server models to inference tasks.

In this paper, we propose a novel metaheuristic algorithm for addressing the challenge of inference task scheduling and offloading in edge computing environments under time and energy constraints. Our primary objective is to devise an efficient algorithm which offers decision-making strategies that dynamically assign inference tasks between local edge models and centralized edge server models, aiming to maximize accuracy while respecting real-time requirements and energy limitations. Through a combination of theoretical analysis, algorithm design, and empirical evaluations, we seek to provide a practical solution that optimizes the allocation of computational resources in edge computing systems, thereby advancing the state-of-the-art in edge computing research and facilitating the development of intelligent and energy-efficient edge applications.

The main contributions of this paper can be summarized as follows:

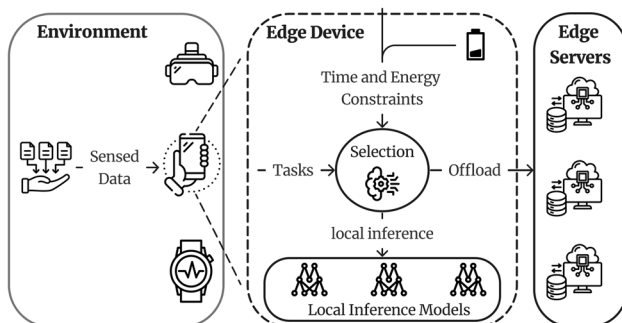


Fig. 1 Inference model selection between local models and the edge server

- We formulate the problem of inference model scheduling and offloading in parallel under time and energy constraints by providing an accuracy, inference time and energy models and propose an optimization problem.
- We propose HGSTO a hybrid genetic algorithm (GA) leveraging the accuracy of GA and the speed of Simulated Annealing (SA) which help accelerate the evolution process and converge to the best solution in fewer generations.
- We perform experiments and compare the performance of HGSTO against GA, SA, and Particle Swarm Optimization (PSO) where we prove the efficiency and accuracy of HGSTO.

The rest of this paper is organized as follows. Section 2 presents the related works. In Sect. 3 we describe the system model. In Sect. 4 we propose HGSTO and explain the solution steps. In Sect. 5 we present the experiment setup and results in addition to analysis of the obtained results. Finally, we conclude this work in Sect. 6.

2 Related works

The problem of task offloading in edge computing has garnered significant attention in recent literature due to its crucial role in optimizing resource utilization and enhancing system performance. Various approaches have been proposed to address different aspects of task offloading [14]. For instance, works such as [15–17] focus on offloading decisions based on data characteristics and network conditions, aiming to minimize latency and maximize throughput. Similarly, [18–20] explore task offloading strategies considering energy consumption and device capabilities, with a focus on energy-aware scheduling algorithms. Other works, such as [21–23] targeted both energy and latency as an objective where the aim to reduce both latency and energy or strike a compromise between them. These works however, are not suitable for real-time applications where strict deadlines and energy budgets must be met.

Few works have explicitly considered time and energy constraints, either individually or in combination. For instance, [24–26] propose a time-aware task offloading framework that prioritizes tasks based on their deadlines and dynamically allocates resources to meet real-time requirements. Similarly, [27, 28] introduce an energy-aware task offloading approach that optimizes energy consumption by balancing workload distribution across edge devices and servers. Furthermore, [29–31] investigate the joint optimization of time and energy constraints in task offloading decisions, employing mathematical modeling

techniques to formulate the problem as a multi-objective optimization task. These works underscore the importance of considering both time and energy constraints in task offloading decisions, highlighting their interdependence and the need for holistic optimization approaches in edge computing environments.

Various methods have been employed to address the task offloading problem in edge computing, reflecting the diverse nature of the challenges involved. Traditional optimization techniques, such as mixed linear programming (MLP) [32, 33] and branch and bound [33], have been utilized to formulate task allocation as a mathematical optimization problem, enabling the derivation of exact or near-optimal solutions under specific constraints. Additionally, machine learning approaches have gained prominence for their ability to adaptively learn and optimize task offloading decisions based on historical data and real-time observations [34, 35]. Furthermore, metaheuristic algorithms, including genetic algorithms, simulated annealing, and particle swarm optimization, have been applied to tackle the NP-hard nature of the task offloading problem by efficiently exploring the solution space and identifying high-quality solutions [36, 37]. Each of these methods offers distinct advantages and trade-offs, depending on the specific characteristics of the problem instance and the requirements of the edge computing environment.

The only two works in the literature that try to improve the inference accuracy of an edge computing system through offloading to edge servers, are the work by [12], where the authors propose a data selection scheme for IoT devices in which an edge device can decide to offload data that would likely lead to inaccurate inference if processed locally and thus improving the overall accuracy of the whole system. Their data scheme performs the selection under a given energy constraint. The proposed scheme shows promising results, however, it does not consider the time constraint and thus renders it unsuitable for real time applications. In addition, their scheme only considers a single inference model in the edge device which does not offer many options in terms of maximizing accuracy. On the other hand, the authors of [13] studied the inference task offloading under a time constraint in a system where edge devices are equipped with multiple inference models varying in size and accuracy in addition to an edge server. Their system leverages the fact that they have two parallel machines namely the edge device and the edge server. They proposed AMR^2 , a scheduling scheme based on LP-Relaxation and rounding which considers all possible cases of scheduling two tasks between the edge device and the edge server. They relax the problem's constraint to take fractional values and then perform rounding to get the result. In the edge device they

use dynamic programming to schedule the tasks. Their proposed scheme performed better than the greedy approach. However, their system does not take into consideration the energy constraint of the edge device. Although, using the edge device’s onboard inference models along with offloading tasks to the edge server can reduce the total inference time of the system, in the case of battery-powered edge devices, it costs a significant amount of valuable energy. These works leave a research opportunity for the more general and practical problem of considering both time and energy constraints while leveraging the dynamicity of having multiple local inference models in addition to multiple edge servers. This problem is significantly more challenging however it better represents real-world applications.

As far as we know, no other work has considered the problem of inference task offloading to maximize the accuracy under both time and energy constraints. Therefore, this work is designed to fill this research gap by providing a practical solution to this problem.

3 System model

Considering a system (see Fig. 2) where each edge device is equipped with a set of local inference models denoted as $\mathcal{L} = \{1, \dots, L\}$. Every edge node is connected to a set of edge servers \mathcal{E} each equipped with a single inference model. The set of server inference models is denoted as $\mathcal{S} = \{1, \dots, S\}$. An edge node have access to a set of inference models $\mathcal{M} = \{1, \dots, M\} = \mathcal{S} \cup \mathcal{L}$ consisting of both \mathcal{S} and \mathcal{L} . At each time slot the edge node receives a set of inference tasks denoted as $\mathcal{J} = \{1, \dots, J\}$.

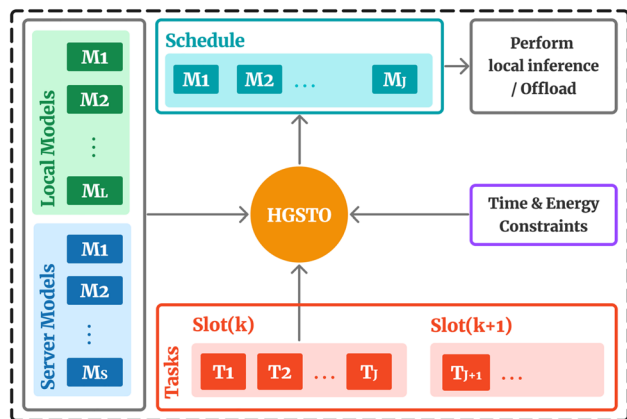


Fig. 2 HGSTO system model (Color figure online)

3.1 Inference accuracy

The inference models deployed in edge nodes can be a single tunnable model where adjusting the input hyperparameters changes the accuracy and inference times. Another option is to deploy multiple instances of a similar type models with different internal structures such as layer sizes in the case of Deep Neural Networks (DNNs). Alternatively, different types of models varying in size and top-1 average accuracy also can be used. Since the real top-1 accuracy of each model for a given inference task is not known prior to performing the inference we use the average accuracy estimated over historical measured top-1 accuracy. The average top-1 accuracy of model i is denoted as A_i where $i \in \mathcal{M}$. The average top-1 accuracy of edge server models is set to be significantly higher than edge device local inference models.

$$A_j > A_i \quad \forall i \in \mathcal{L}, j \in \mathcal{S}$$

3.2 Time delay model

Similar to the estimated average accuracy we estimate average inference time for each model i denoted as T_i^{inf} where $i \in \mathcal{M}$ obtained by averaging historical measured inference times. Data preprocessing time is considered to be part of T_i^{inf} .

Let T_i^{lat} be the average communication latency of edge server i which can be estimated from historical measured data and is continuously updated after every transmission. We define T_{ij}^{off} to be the estimated time to offload task j where $i \in \mathcal{J}$ to edge server i . T_{ij}^{off} can be estimated from the connection bandwidth and the size of task j denoted as $size_j$. T_{ij}^{off} is given by:

$$T_{ij}^{off} = \frac{size_j}{b_i} + T_i^{lat}$$

where b_i is the bandwidth of the communication channel between the edge device and edge server i . The value of b_i can be measured and updated regularly by probing the edge server.

We define T_{ij}^{task} as the total time it takes to process task j using model i including inference and offloading times.

$$T_{ij}^{task} = T_i^{inf} \quad \forall i \in \mathcal{L} \quad j \in \mathcal{J}$$

$$T_{ij}^{task} = T_i^{inf} + T_{ij}^{off} + T_i^{resp} \quad \forall i \in \mathcal{S} \quad j \in \mathcal{J}$$

where T_i^{resp} represents the average response time from edge server i given by:

$$T_i^{resp} = \frac{size_r}{b_i} + T_i^{lat}$$

Table 1 Symbols and notations

Notation	Description
\mathcal{L}	The set of edge device local inference models
\mathcal{E}	The set of available edge servers
\mathcal{S}	The set of edge server inference models
\mathcal{M}	The set of all inference models available to the edge device
\mathcal{J}	The set of inference tasks for each time slot
A_i	The average top-1 accuracy of model i
T_i^{inf}	The Average inference time of model i
T_i^{lat}	The average communication latency for edge server i
T_{ij}^{off}	The estimated time to offload task j to the edge server i
$size_j$	The size of task j
b_i	The bandwidth of the communication channel for edge server i
T_{ij}^{task}	The total time it takes to process task j using model i
T_i^{resp}	The average response time from edge server i
x_{ij}	Represents whether an inference model i is assigned to task j
T_k^{slot}	The total time to process a complete time slot k
T_k^{local}	The total local inference time
T_k^{server}	The total server time for time slot k
$T^{\text{off_accu}}$	Offloading time accumulator
$T^{\text{inf_accu}}$	Inference time accumulator
$E_i^{\text{off } j}$	The energy cost of offloading task j to edge server i
c_i	The average energy cost of transmitting data to edge server i
E_i^{resp}	The energy cost of the inference response
E_i^{inf}	The average energy cost of inference using model i
$E_i^{\text{task } j}$	The total energy cost of processing task j using model i
A_k^{slot}	The total accuracy for time slot k
E_k^{slot}	The total energy for time slot k
\mathcal{P}	Population of solutions
s	solution or schedule
f	Fitness function
N	Number of past generations to check for termination condition
p_{accept}	Acceptance probability
μ	Mutation probability
γ	Mutation probability fading parameter
ϕ	Number of elite individuals passed to next generation
ϵ	Minimum fitness gradient to consider termination condition
ψ	Tournament selection size

where $size_r$ is a constant representing the response size.

Let $x_{ij} = \{0, 1\}$ be a variable representing whether an inference model i is assigned to inference task j . We define T_k^{slot} as the total time to process a complete time slot $k \in \Delta$. Since the local inference and offloading are performed in

parallel, we define T_k^{slot} as the max between the total local inference time denoted as T_k^{local} and the total server time which includes offloading, inference and response times of all offloaded tasks denoted as T_k^{server} .

$$T_k^{\text{slot}} = \max(T_k^{\text{local}}, T_k^{\text{server}}) \quad \forall k \in \Delta$$

where

$$T_k^{\text{local}} = \sum_{i=1}^L \sum_{j=1}^J x_{ij} t_{ij}$$

T_k^{server} is calculated using Algorithm 1.

Algorithm 1 Steps to calculate T_k^{server}

```

 $T^{\text{off\_accu}} = 0$ 
for  $j$  in  $\mathcal{J}$ :
  for  $i$  in  $\mathcal{M}$ :
     $T^{\text{off\_accu}} = T^{\text{off\_accu}} + x_{ij} T_{ij}^{\text{off}}$ 
     $T_i^{\text{inf\_accu}} = \max(T_i^{\text{inf\_accu}}, T_{ij}^{\text{off\_accu}}) + x_{ij} T_i^{\text{inf}}$ 
     $T_i^{\text{server}} = \max(T_i^{\text{server}}, T_i^{\text{inf\_accu}}) + x_{ij} T_i^{\text{resp}}$ 
   $T_k^{\text{server}} = \max(T_i^{\text{server}}) \quad \forall i \in \mathcal{M}$ 

```

T_i^{server} is the total offload, inference and response times for all tasks offloaded to edge server with model i . $T^{\text{off_accu}}$ is a variable to accumulate and track offload times for all edge servers. $T_i^{\text{inf_accu}}$ is a variable that accumulates and tracks inference times for each edge server with model i .

The edge devices including normal edge nodes and edge servers, are assumed to have two queues as depicted in Fig. 3, one for performing computation and another for communication. This allows for parallel inference and offloading or receiving. In Fig. 4, we show an example of a schedule for 10 inference tasks where 4 tasks are processed using local inference models and 6 are offloaded to 3 different edge servers. At t_1 we have:

$$T^{\text{off_accu}} = T_{1,1}^{\text{off}}$$

$$T_i^{\text{inf_accu}} = \max(T_i^{\text{inf_accu}}, T^{\text{off_accu}}) + T_{1,1}^{\text{inf}}$$

$$T_i^{\text{inf_accu}} = T^{\text{off_accu}} + T_{1,1}^{\text{inf}} = T_{1,1}^{\text{off}} + T_{1,1}^{\text{inf}}$$

$$T_i^{\text{server}} = \max(T_i^{\text{server}}, T_i^{\text{inf_accu}}) + T_1^{\text{resp}}$$

$$T_1^{\text{server}} = T_i^{\text{inf_accu}} + T_{1,1}^{\text{resp}} = T_{1,1}^{\text{off}} + T_{1,1}^{\text{inf}} + T_{1,1}^{\text{resp}}$$

Following the steps in Algorithm 1 on the example shown in Fig. 4, we find T_i^{server} values shown in Table 2. The value of T^{server} is taken as the maximum value of all edge server total times. Finally, the total slot time T^{slot} is found by taking the larger value between total time of local inference T^{local} and T^{server} .

$$T^{\text{server}} = \max(T_1^{\text{server}}, T_2^{\text{server}}, T_3^{\text{server}}) = T_2^{\text{server}}$$

$$T^{\text{slot}} = \max(T^{\text{local}}, T_2^{\text{server}}) = T_2^{\text{server}}$$

3.3 Inference energy

Let E_{ij}^{off} be the energy cost of offloading task j to edge server i . E_{ij}^{off} depends on the offload time T_{ij}^{off} and c_i the average energy cost of transmitting data to edge server i in one time unit. c_i depends on several factors, including the communication medium such as Wi-Fi, Cellular, Bluetooth, or Zigbee, affects energy consumption. Each medium has different power requirements, data rates, and transmission ranges, which influence the overall energy cost. The energy consumption of a wireless device is significantly influenced by its transmission power level. Typically, higher transmit power levels lead to increased energy usage, particularly when maintaining communication over longer distances or in challenging environments with obstacles or interference. Additionally, the power consumed by the device while idle or in standby mode adds to the overall energy cost. Furthermore, signal strength and quality play crucial roles in determining energy consumption, particularly in systems where transmission power adapts based on signal conditions. In scenarios where reliable communication is essential, such as environments with weak or noisy signals, higher power levels may be necessary, resulting in heightened energy consumption. Moreover, environmental factors like interference, obstacles, and electromagnetic noise can further impact energy consumption by influencing signal propagation and reception quality. Various common optimization techniques are must also be taken into consideration such as data compression, packet aggregation, adaptive modulation, and power control algorithms To mitigate energy consumption during wireless communication which significantly impact energy consumption.

We assume that c_i can be calculated internally by monitoring battery usage and the network adapter’s configurations such as the transmission power. By averaging

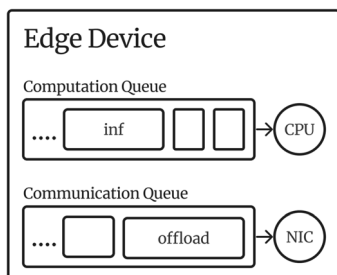


Fig. 3 Edge device model

these measured power usage metrics we can estimate c_i . E_{ij}^{off} is given by:

$$E_{ij}^{\text{off}} = \frac{T_{ij}^{\text{off}}}{c_i}$$

Similarly, the energy cost of the inference response denoted by E_i^{resp} is given by:

$$E_i^{\text{resp}} = \frac{T_i^{\text{resp}}}{c_i}$$

Let E_i^{inf} be the average energy cost of performing the inference of an inference task using model i . The inference energy cost is negligible compared to the offloading energy cost. Therefore it is defined as a constant which can be estimated using the inference time and the maximum power consumption of the edge device’s CPU in the worst case under full load. Let E_{ij}^{task} be the total energy cost of processing task j using model i given by:

$$E_{ij}^{\text{task}} = E_i^{\text{inf}} \quad \forall i \in \mathcal{L} \quad j \in \mathcal{J}$$

$$E_{ij}^{\text{task}} = E_i^{\text{inf}} + E_{ij}^{\text{off}} + E_i^{\text{resp}} \quad \forall i \in \mathcal{S} \quad j \in \mathcal{J}$$

Finally, we define E_k^{slot} as the total energy consumption for slot k given by:

$$E_k^{\text{slot}} = \sum_{i=1}^M \sum_{j=1}^J x_{ij} E_{ij}^{\text{task}}$$

3.4 Problem formulation

In this section, we model the inference models allocation for inference tasks under time and energy constraints while maximizing accuracy as a single objective optimization problem. The optimization problem is formulated as follows:

$$\text{Maximize } A_k^{\text{slot}} = \sum_{i=1}^M \sum_{j=1}^J x_{ij} A_i \tag{1}$$

Where A_k^{slot} is the total accuracy for a time slot k . Let E_k^{slot} be the total energy consumption of slot k . Given E and T as the energy and time constraints respectively, Eq. (1) is subject to:

$$T_k^{\text{slot}} \leq T \quad \forall k \in \Delta \tag{2}$$

$$E_k^{\text{slot}} \leq E \quad \forall k \in \Delta \tag{3}$$

$$\sum_{i=1}^M x_{ij} = 1 \quad \forall j \in \mathcal{J} \tag{4}$$

Using Eq. (2) we guarantee each parallel processing time of each slot is respecting the time constraint. Similarly,

Fig. 4 An example of a schedule for 3 edge servers and 10 inference tasks (Color figure online)

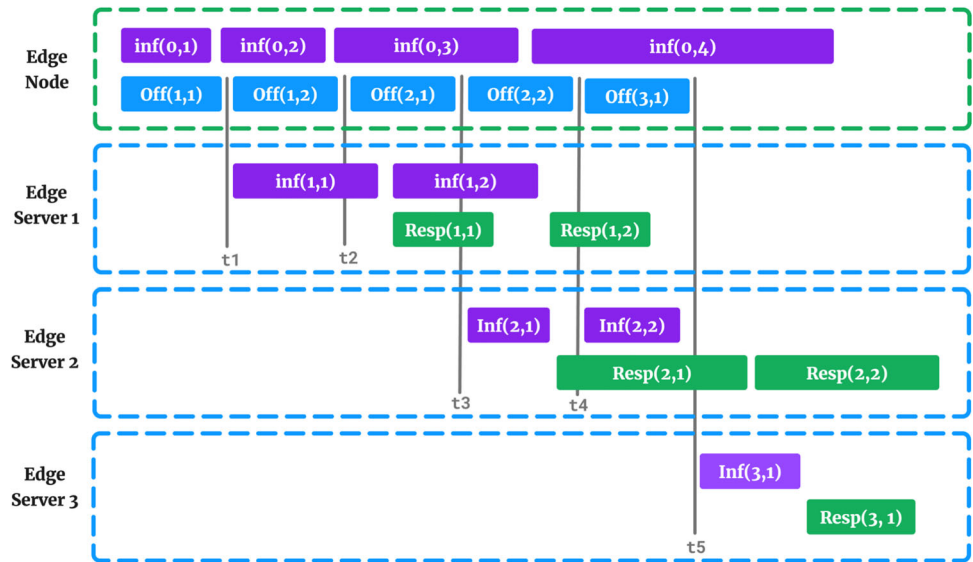


Table 2 Example values of T_i^{server} for time steps 2–5

t_2	$T_1^{server} = T_{1,1}^{off} + T_{1,1}^{inf} + T_{1,2}^{inf} + T_{1,2}^{resp}$
t_3	$T_2^{server} = T_{1,1}^{off} + T_{1,2}^{off} + T_{2,1}^{off} + T_{2,1}^{inf} + T_{2,1}^{resp}$
t_4	$T_2^{server} = T_{1,1}^{off} + T_{1,2}^{off} + T_{2,1}^{off} + T_{2,1}^{inf} + T_{2,1}^{resp} + T_{2,2}^{resp}$
t_5	$T_3^{server} = T_{1,1}^{off} + T_{1,2}^{off} + T_{2,1}^{off} + T_{2,2}^{off} + T_{3,1}^{off} + T_{3,1}^{inf} + T_{3,1}^{resp}$

Eq. (3) ensures energy consumption for a time slot respects the given energy constraint. Finally, Eq. (4) guarantees each inference task is assigned an inference model which produces a complete solution.

This problem could be thought of as an instance of the well known classic knapsack problem (KP). In which, we are trying to fill our schedule (i.e. a knapsack) with inference models (i.e. pieces) to maximize the accuracy (i.e. profit) while respecting time and energy constraints (i.e. knapsack weight and volume capacities). In this case, the pieces and the knapsack have 2-dimensions and therefore this problem is an instance of the multi-dimensional KP. Additionally, inference models (i.e. pieces) can be reused to construct a schedule therefore the problem becomes an instance of the unbounded multi-dimensional KP (UMdKP). However, since we are considering parallel schedules where inference tasks can be processed locally and in edge servers in parallel, which in UMdKP terms means pieces can be overlapping in the weight dimension (i.e. time) but not in the volume dimension which renders this similarity useless. Alternatively, multiple knapsacks can be considered for each edge server which makes this problem an instance of the multi-knapsack problem (MKP). However, the problem becomes much more

difficult to model specially when trying to uphold the time constraint over all knapsacks.

4 A hybrid genetic algorithm for selective inference task offloading (HGSTO)

In this section we propose a hybrid genetic algorithm for selective task offloading (HGSTO) using genetic algorithms (GA) and simulated annealing (SA). The rationale for incorporating GAs into our methodology stems from their inherent capability to efficiently explore complex, multidimensional solution spaces, particularly in optimization problems characterized by diverse and interrelated variables. Despite their effectiveness in traversing vast search spaces, GAs are susceptible to convergence towards local minima, potentially limiting the discovery of globally optimal solutions. To mitigate this limitation, we augment the GA framework with a simulated annealing local search step. Simulated annealing, renowned for its ability to escape local optima through probabilistic transitions, offers a complementary mechanism to the deterministic exploration of GAs. By incorporating SA as a local search mechanism within the GA framework, we enhance the algorithm’s ability to escape local minima while accelerating the evolution process. This synergistic combination harnesses the accuracy of GAs for multidimensional problems while leveraging the speed and robustness of SA to prevent premature convergence and promote the discovery of high-quality solutions [38, 39]. The main steps of HGSTO are illustrated in Fig. 5.

4.1 Population initialization

Considering a population of solutions (i.e., schedules) \mathcal{P} of size P where each solution s is a vector of model indices corresponding to each task given by $s = \{i, \forall i \in \mathcal{M}\}$ such that $|s| = |\mathcal{J}|$. The initial population can greatly impact the speed of convergence of the genetic algorithm where higher coverage of the solution space is critical to finding the best solution. Furthermore, increasing the population density in promising areas can also accelerate the convergence of solutions. However, in this work our main goal is to design a fast algorithm which renders complex initialization schemes unsuitable options. We experimented with multiple initialization schemes including the Latin hypercube sampling and uniform random initialization and found that these schemes only add runtime overhead while not providing any improvements to either the accuracy or the convergence speed of the genetic algorithm, therefore in our solution we favor using a simple pseudo-random numbers generator based on a modified version of Donald E. Knuth’s subtractive random number generator algorithm [40].

4.2 Fitness evaluation

The fitness function f of a solution s_k for a time slot k is calculated using T_k^{slot} , A_k^{slot} and E_k^{slot} in addition to time and energy constraints denoted as T and E respectively. We define $f(s_k)$ as follows:

$$\delta_T = \frac{T_k^{\text{slot}} - T}{T}$$

$$\delta_E = \frac{E_k^{\text{slot}} - E}{E}$$

$$\omega_T = \begin{cases} 1 & \text{if } (\delta_T < 0) \text{ and } (|\delta_E| \leq \delta_{\text{min}}) \\ 1 - |\delta_T| & \text{otherwise} \end{cases} \quad (5)$$

$$\omega_E = \begin{cases} 1 & \text{if } (\delta_E < 0) \text{ and } (|\delta_T| \leq \delta_{\text{min}}) \\ 1 - |\delta_E| & \text{otherwise} \end{cases} \quad (6)$$

$$\omega = \frac{1}{2}\omega_T + \frac{1}{2}\omega_E \quad (7)$$

$$f(s_k) = \omega A_k^{\text{slot}} \quad (8)$$

δ_T and δ_E represent the distance ratio of T_k^{slot} and E_k^{slot} from the given constraints T and E respectively. δ_{min} is the minimum distance ratio where we consider the constraint is met. ω_T and ω_E are the time and energy penalties respectively. The penalties are designed such that they scale with the distance (either larger or smaller) from the given constraints which forces the solution to minimize the distance and thus take advantage of the given constraints budget. However, in some cases we can face a limitation

from a single constraint in which Eqs. (5) and (6) ensure that we do not force the solutions to maximize both constraints by dropping the penalty for the other constraint as long as it is respected.

4.3 Termination condition

The termination condition of a GA is a critical stopping criterion that determines when the optimization process should stop. Common termination conditions include reaching a specified number of max generations, achieving a desired level of fitness improvement, or surpassing a predefined computational budget. Choosing the appropriate termination condition is crucial to the performance of HGSTO. Therefore, we use a solution stagnation monitoring method to halt the evolution when the algorithm fails to produce significantly improved solutions over a certain number of generations, indicating that further iterations are unlikely to yield substantial improvements. By monitoring solution stagnation, the GA can dynamically adjust its exploration-exploitation trade-off, allocating computational resources more efficiently towards promising regions of the search space. This termination condition helps prevent the algorithm from wasting computational resources on unproductive iterations, thereby improving convergence speed and solution quality. The fitness of the top-1 ranking solution of each generation is compared against solutions from the last N generations to determine whether there is

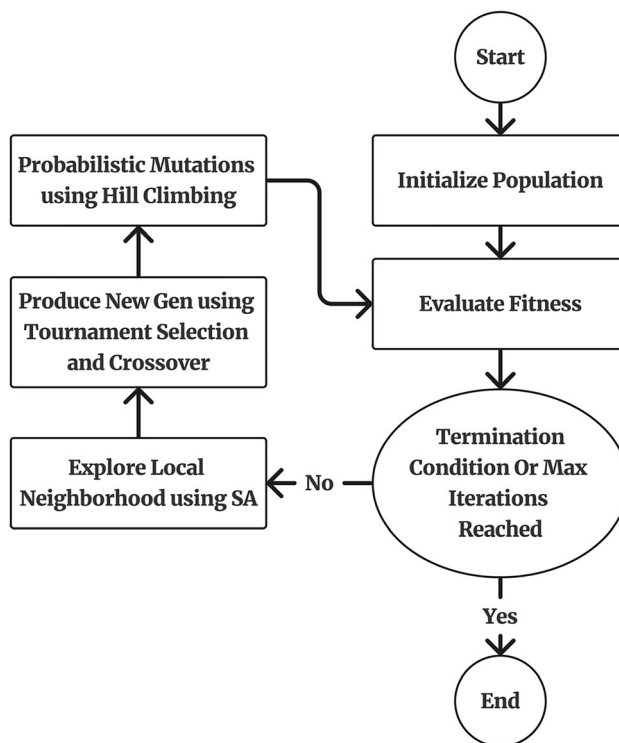


Fig. 5 HGSTO steps flow diagram

any significant improvement [41]. The termination condition steps are presented in Algorithm 2.

Algorithm 2 Termination condition

```

func termination_condition(g, fittest):
    f' ← fittest[g − N]
    for s, f ∈ fittest[gen − N + 1...]:
        if (f − f') > ε:
            return False
        f' ← f
    return True:

```

4.4 Neighborhood exploration

GAs are known to be susceptible to falling into local minima due to their reliance on deterministic mechanisms for solution exploration and exploitation. As GAs iteratively evolve a population of candidate solutions through processes such as selection, crossover, and mutation, they tend to converge towards regions of the search space that offer locally optimal solutions. However, this deterministic exploration can inadvertently lead to premature convergence, trapping the algorithm in suboptimal solutions known as local minima. To mitigate this limitation, incorporating simulated annealing as a local search (i.e., neighborhood exploration) method offers a complementary approach. SA introduces stochasticity into the optimization process, allowing the algorithm to explore regions beyond local optima with a certain probability, thereby escaping local minima. By leveraging SA's probabilistic transitions, GAs can effectively navigate complex solution spaces, balancing exploration and exploitation to promote the discovery of globally optimal solutions. This synergistic combination of GA and SA not only enhances the algorithm's robustness against local minima but also accelerates convergence towards high-quality solutions in challenging optimization problems [39].

Simulated annealing (SA) is an optimization technique inspired by the physical process of annealing in metallurgy, where a material is heated and then slowly cooled to reach a low-energy state. In the context of optimization, SA iteratively explores the solution space to find the optimal solution by mimicking this annealing process. The steps of simulated annealing are presented in Algorithm 4. It starts by initializing the algorithm with an initial solution and setting parameters such as the initial temperature t_0 and cooling rate c_r . The initial temperature controls the level of exploration in the early stages of the algorithm, while the cooling rate determines how the temperature decreases over iterations. At each iteration, we generate a neighboring solution s_{new} by applying a perturbation to the current solution as shown in Algorithm 3. This perturbation could

involve randomly modifying one or more components of the solution, such as swapping two elements or perturbing values within a certain range. s_{new} is then evaluated and compared against the current solution $s_{current}$ to determine whether to accept it as the current solution if its fitness value $f(s_{new})$ is better than $f(s_{current})$. Otherwise, if the neighboring solution is worse, accept it with a probability given by Eq. (9) which is based on the difference in objective function values and the current temperature. The temperature is then updated according to the cooling rate c_r which controls the rate at which the algorithm transitions from exploration to exploitation, gradually reducing the likelihood of accepting worse solutions as the algorithm progresses [42].

Algorithm 3 SA get neighbor algorithm

```

func get_neighbor(s):
    snew ← s
    r ← random(0, |s|)
    step ← random({−1, 1})
    snew[r] ← min(max(snew[r] + step, 0), |M| − 1)
    return snew

```

Algorithm 4 SA algorithm

```

func sim_annealing(s, max_iter, t0, crate):
    t ← t0
    sbest ← s
    scurrent ← s
    for i = 1 to max_iter:
        snew ← get_neighbor(scurrent)
        eold ← f(scurrent)
        enew ← f(snew)
        if Paccept(eold, enew, t) > random():
            scurrent ← snew
        if enew > f(sbest):
            sbest = snew
        t ← t(1 − crate)
    endfor
    return sbest

```

$$P_{\text{accept}}(e_{\text{old}}, e_{\text{new}}, t) = \begin{cases} 1.0 & \text{if } (e_{\text{new}} > e_{\text{old}}) \\ e^{\frac{e_{\text{new}} - e_{\text{old}}}{t}} & \text{otherwise} \end{cases} \quad (9)$$

4.5 Reproduction process

A crossover operator plays a crucial role in generating new offspring solutions by combining information from two parent solutions. One common approach is the uniform crossover operator, where the offspring solution o is constructed by randomly selecting values from each parent

solution with equal probability. In this process, for each inference model index in the offspring solution, a random choice is made between the corresponding values from the two parents p_1 and p_2 . This crossover mechanism promotes diversity in the offspring population by recombining genetic information from different parent solutions, potentially leading to the discovery of novel and high-quality solutions in the search space. The parents are selected from the population using tournament selection, where a number ψ of individuals, referred to as tournament size, are randomly chosen from the population. These individuals then compete against each other, and the one with the highest fitness value is selected as a parent. This process is repeated until the desired number of parents for mating is reached. Tournament selection offers several advantages, including simplicity of implementation, computational efficiency, and robustness against premature convergence. By allowing weaker individuals to participate in the tournament, tournament selection maintains diversity in the population, enabling the algorithm to explore a wider range of solutions. Additionally, the tournament size parameter allows for control over the selection pressure, with larger tournaments favoring stronger individuals and smaller tournaments promoting diversity.

The newly generated offspring undergo probabilistic mutations to keep the population diverse and help nudge solutions out of local minima. A mutation probability parameter δ is introduced to determine whether an offspring undergoes a mutation. A higher mutation probability increases the chances of introducing more random changes, which can be beneficial for exploration but may also disrupt promising solutions. Conversely, a lower mutation probability might lead to slower exploration but can help preserve promising solutions. Using a fading parameter γ to reduce the mutation probability δ over iterations, helps to initially promote exploration with a higher mutation rate to discover diverse regions of the solution space and gradually decrease it to allow for exploitation and refinement of promising solutions.

Instead of applying random mutations to the offspring, we use a hill climbing algorithm (see Algorithm 6) which guarantees that the mutation is guiding the offspring solutions out of any local minima towards the global optimal solution. This helps further accelerate the evolution of solutions to converge in fewer iterations.

We employ an elitism mechanism to preserve the best solutions from one generation to the next. In elitism, a number φ of the fittest individuals from the current generation are directly carried over to the next generation without undergoing any genetic operations, such as crossover or mutation. This ensures that the best solutions found so far are not lost during the evolution process which in turn accelerates the evolution process.

Putting all steps together we provide HGSTO's pseudo code in Algorithm 5.

Algorithm 5 HGSTO algorithm

```

 $\mathcal{P} \leftarrow \text{init\_population}()$ 
 $\text{fittest} \leftarrow \{\}$ 
for  $g = 1$  to  $G$ :
   $\text{ranked} \leftarrow \{\}$ 
  for  $s \in \mathcal{P}$ :
     $\text{ranked} \leftarrow \text{ranked} + (s, f(s))$ 
   $\text{sort}(\text{ranked})$ 
   $\text{ranked} \leftarrow \{s_i \mid \forall s_i \in \text{ranked}, i \in [0, P]\}$ 
   $\text{ranked}[0] \leftarrow \text{sim\_annealing}(\text{ranked}[0])$ 
   $\text{fittest} \leftarrow \text{fittest} + \text{ranked}[0]$ 
  if  $g > 0$  and  $g \% N = 0$ :
    if  $\text{termination\_condition}(g, \text{fittest})$ :
      break
   $\text{new\_gen} \leftarrow \{s_i \mid \forall s_i \in \text{ranked}, i \in [0, \varphi]\}$ 
  for  $j = 1$  to  $P - \varphi$ :
     $p_1 \leftarrow \text{tournament\_select}(\text{ranked}, \psi)$ 
     $p_2 \leftarrow \text{tournament\_select}(\text{ranked}, \psi)$ 
     $o \leftarrow \text{crossover}(p_1, p_2)$ 
    if  $\text{random}() \leq \mu$ :
       $\text{mutate}(o)$ 
     $\text{new\_gen} \leftarrow \text{new\_gen} + \{o\}$ 
   $\mathcal{P} \leftarrow \text{new\_gen}$ 
   $\mu \leftarrow \mu \times \gamma$ 
 $\text{sort}(\text{fittest})$ 
return  $\text{fittest}[0]$ 

```

Algorithm 6 Hill climbing algorithm

```

func  $\text{hill\_climbing}(s_0, \text{step}, H)$ :
   $s_{\text{current}} \leftarrow s_0$ 
  for  $i = 1$  to  $H$ :
     $s_{\text{neighbor}} \leftarrow s_{\text{current}} + \text{random}(\{-\text{step}, 0, \text{step}\})$ 
    if  $f(s_{\text{neighbor}}) > f(s_{\text{current}})$ :
       $s_{\text{current}} \leftarrow s_{\text{neighbor}}$ 
  return  $s_{\text{current}}$ 

```

4.6 Complexity analysis of HGSTO

Assuming the fitness function has a time complexity of $O(F)$ for a single solution, we can analyse the impact of evaluating the fitness function on the overall time complexity of the GA with SA as a local search by finding the complexity of each algorithm separately. Let G be the number of generations, P be the population size, J be number of inference model indexes per solutions, which is

also the number of inference tasks per time slot, Q be the number of iterations in SA, F be the time complexity of the fitness function for a single solution.

Genetic algorithm operations: Selection, crossover, and mutation: $O(P \times J)$ per generation. Fitness evaluation: $O(P \times F)$ per generation. SA is performed once per generation with A iterations. Each iteration involves generating a neighboring solution, evaluating its fitness, and accepting or rejecting it based on a probabilistic criterion. Considering A iterations, the time complexity of SA per generation is approximately $O(A \times (J + F))$.

Combining the time complexities of genetic algorithm operations and simulated annealing, the overall time complexity of the GA with SA as a local search is approximately $O(G \times (P \times F + A \times (J + F)))$.

It is important to note that this is the time complexity of the worst case scenario in which the termination criterion is never used. In practice and in most cases, HGSTO converges before reaching G generations.

5 Experimental results

In this section, we assess the performance of the proposed HGSTO algorithm through testbed experiments. The following algorithms are examined as reference points.

First, Particle Swarm Optimization (PSO) is one of the classic metaheuristics used in the literature for the MdKP [43, 44], particularly renowned for its efficacy in exploring high-dimensional and complex search spaces which makes it a suitable baseline method. PSO mimics the social behavior of bird flocks or fish schools, where individuals, known as particles, navigate the search space by iteratively adjusting their positions based on their own experiences and the collective knowledge of the swarm. This cooperative search mechanism enables PSO to efficiently explore diverse regions of the search space, adapt to dynamic environments, and converge towards optimal solutions. By comparing the performance of our proposed algorithm against PSO, we can assess its effectiveness and competitiveness in addressing the optimization problem at hand. PSO in its original form is primarily designed for searching continuous search spaces, where solutions are represented by real-valued vectors. However, in our experiments, we adapt PSO to effectively handle discrete search spaces by using integer position vectors representing inference model indexes. In addition, to integer velocity vectors where after every update to the position vector we perform clipping on the new values to make sure they stay in the permitted range.

Second, simulated annealing (SA) has been used in the literature for both task scheduling [45, 46] and the MdKP [47, 48] and proven to be effective. Additionally, its

stochastic nature in which it uses a probabilistic acceptance criterion that allows it to escape local optima and explore diverse regions of the search space, making it particularly well-suited for handling complex and multimodal optimization problems and a compelling baseline choice. Furthermore, since HGSTO uses SA as step to accelerate convergence, it makes it important to see a direct comparison of HGSTO against the standalone SA. SA is implemented similar to HGSTO following Algorithms 4 and 3.

Finally, we compare against the unmodified version of GA to observe the improvements HGSTO provides on top of GA. GA is implemented similar to HGSTO with no additional modifications. All compared algorithms including HGSTO use the same fitness function given in Eq. (8). Parameters for all implemented algorithms are presented in Table 4. The language used for implementing all algorithms is Python version 3.12, in addition to PyTorch for inference models. Parameters such as population size and swarm size for HGSTO, GA and PSO have been set to the minimum values that offer the highest performance, higher values than these only increase execution times while providing no improvement to performance.

We conduct a real-world case study where we use a Raspberry Pi 5 as the edge device, which is equipped with a quad-core Arm Cortex A76 processor @ 2.4GHz and 8GB of RAM. The edge device is connected to an access point through WiFi (802.11ac) which has a set of edge servers consisting of desktop computers connected by Ethernet. In showcasing the efficacy of the proposed algorithm for real-world scenarios, it is important to highlight that, to mitigate potential traffic contention with other devices, the experiments were carried out in an environment devoid of concurrent devices competing for network resources. In our experiments, we use power as a constraint measured in watts instead of energy measured in kWh for convenience.

The case study consists of an object detection application using the ImageNet-Mini dataset [49] as a source of data with 3923 images ranging in size from 10KB to 10MB. The edge device is assumed to receive 10 inference tasks (i.e., images) per time slot. We deploy three lightweight object classification models namely ResNet18 and ResNet34 [50], in addition to ShuffleNetV2 [51] on the edge device used for local inference. More inference models have been considered such as AlexNet, GoogleNet, MobileNetV2 and SqueezeNet1.1. However, these models did not offer high enough accuracy to inference time ratios (i.e., inefficient) compared to the chosen models which resulted in algorithms never using them in any solutions and thus, they were eliminated from the experiment. In the edge servers we deploy ResNeXt101 [52] which is a larger and more accurate model. During the deployment phase we

run tests on these models to estimate the average inference time and energy consumption of each inference model on each machine (see Table 3).

5.1 Evaluation metrics

We employ four metrics to assess the performance of HGSTO: scheduling time, accuracy, execution time, and power consumption. Scheduling time, quantified in milliseconds, denotes the duration required by the algorithm to generate a schedule for a given time slot. This metric is measured individually for each time slot and subsequently averaged across all slots. Similarly, accuracy, execution time, and power consumption determine the quality and adherence of the schedules generated by the algorithm to the prescribed constraints. Each schedule for every time slot undergoes evaluation through inference utilizing the designated inference models. Subsequently, these metrics are aggregated across all time slots to yield a comprehensive assessment across the dataset.

5.2 Performance under different number of iterations

We initially assess the algorithms' performance across varying numbers of iterations from 10 to 500 iterations in order to determine optimal values for subsequent experiments and ensure fair comparisons. Figure 6 subplot 1, illustrates the scheduling time trends for GA, PSO, and SA, demonstrating a linear increase in scheduling time with distinct slopes relative to the number of iterations. Notably, while SA exhibits a slower increase, it is bound to surpass HGSTO given enough iterations. In contrast, HGSTO

Table 4 Algorithms parameters

Algorithm	Parameter	Value
HGSTO	Mutation probability μ	0.25
	Mutation fading γ	0.90
	Termination count N	10
	Improvement threshold ε	0.5
	Population P	20
	Generations G	20
	Tournament size ψ	5
	Hill climbing iterations H	10
	Number of elites ϕ	3
	Annealing iterations	10
PSO	Swarm size	50
	Iterations	100
SA	Initial temperature t_0	1e3
	Cooling rate c_r	1e-1
	Iterations	500

maintains a consistent scheduling time as a result of its termination condition, which effectively halts the algorithm when no improved solutions are discerned. In Subplot 2, it is evident that HGSTO yields schedules with comparable or superior average accuracy compared to GA, despite the latter's advantage of running for more iterations. Conversely, SA demonstrates an enhancement in accuracy as the number of iterations increases, resulting in schedules with noticeable improvement. In Subplot 3, it becomes apparent that all algorithms are well adhering to the prescribed time constraint of 500 ms, except for SA in which it is observed that with fewer than 200 iterations, schedules consistently exceed the specified constraint, surpassing the 500 ms threshold. However, as the number of iterations surpasses 200, SA begins to generate schedules that conform more closely to the prescribed constraint. Finally, in subplot 4, both HGSTO and GA exhibit optimal utilization of the given power constraint of 30w, leading to superior average accuracy and reduced scheduling times compared to SA and PSO. This superiority stems from HGSTO and GA's ability to discover high-quality solutions that leverage extensive offloading, as opposed to solely relying on local inference. Notably, HGSTO demonstrates slightly higher power consumption than GA, yet it adheres to the power constraint due to the aforementioned reason of optimizing for superior solution quality.

5.3 Performance under different time constraints

In Fig. 7, we analyse the performance of HGSTO under varying time constraints from 200 to 500 ms while the power constraint is set to 30 W. We note that HGSTO exhibits a marginally higher scheduling time in comparison to SA and GA, yet it delivers superior average accuracy relative to PSO, GA, and SA. Subplot 3 indicates that all methods under the specified parameters conform impeccably to the time constraint provided. Moreover, Subplot 4 highlights that HGSTO excels in identifying solutions that maximize utilization of the designated power constraint (i.e., 30 W), consequently yielding enhanced average accuracy.

5.4 Performance under different power constraints

In Fig. 8, we observe the performance of HGSTO while varying the power constraint from 10 to 50 W with the time constraint set to 500 ms. In Subplot 2, HGSTO demonstrates superior accuracy compared to other algorithms. Subplot 3 unveils a consistent behavior across all compared algorithms: at 10 W, schedules are characterized by a 450 ms duration due to the power limitation, leading to reliance

Table 3 Models average accuracies and inference times on the ImageNet-Mini dataset using our HW

	Average accuracy (%)	Average inference time (ms)	Size (mb)	Numberof parameters
ShuffleNetV2 [51]	66.15	19.44	5.3	1,366,792
ResNet18 [50]	72.01	28.07	44.7	11,689,512
ResNet34 [50]	76.79	42.45	83.3	21,797,672
ResNeXt101 [52]	87.05	–	319.3	83,455,272

on local inference over offloading. However, at 15 W, there is a notable reduction in average time to 350 ms, driven by the optimization of solutions transitioning certain tasks from inferior local inference to offloading onto the most optimal edge server with minimal latency and maximal accuracy. This shift exhausts the power budget of the edge device, rendering it idle and unable to utilize the remaining time. Beyond 15 W, a gradual rise in both average time and accuracy ensues as the power budget expands, until it reaches 30 W, where the edge device becomes time-constrained, rendering additional power futile in further enhancing accuracy.

5.5 Performance under varying number of edge servers

In Fig. 9, we present the results of experiments conducted with varying numbers of available edge servers, ranging from 1 to 10, while maintaining constant time and power constraints at 500 ms and 30 W, respectively. In Subplot 2, it is evident that HGSTO capitalizes on the increasing number of edge servers by generating solutions that leverage offloading, resulting in higher accuracy compared to GA, SA, and PSO. Subplot 3 illustrates that GA, and HGSTO provide solutions with the closest average times to

the given time constraints. However, HGSTO is noticeably better with smaller distances between the average time and the given time constraint. Subplot 4 shows that HGSTO is finding solutions with similar power consumption to GA until 5 edge servers, where HGSTO starts finding superior solutions with higher average power as a result of its ability to better search the solution space.

In our research, we employed the t-test as a pivotal statistical method to ascertain the significance of our findings regarding the performance of our proposed algorithm, HGSTO, in comparison to GA, SA, and PSO. Through extensive experimentation conducted over the ImageNet-Mini dataset, iterated 100 times, we meticulously evaluated the fitness and accuracy of solutions produced by each algorithm. The resulting statistical analysis, as depicted in Table 5, revealed that all calculated p-values were below the conventional significance threshold of 0.05. This compelling evidence underscores the statistical significance of our results, affirming that the superiority of HGSTO over its counterparts is not merely coincidental but substantively rooted in the efficacy of our proposed approach.

Fig. 6 Performance comparison under varying iteration numbers with time constraint set to 500 ms and power constraint set to 30 W (Color figure online)

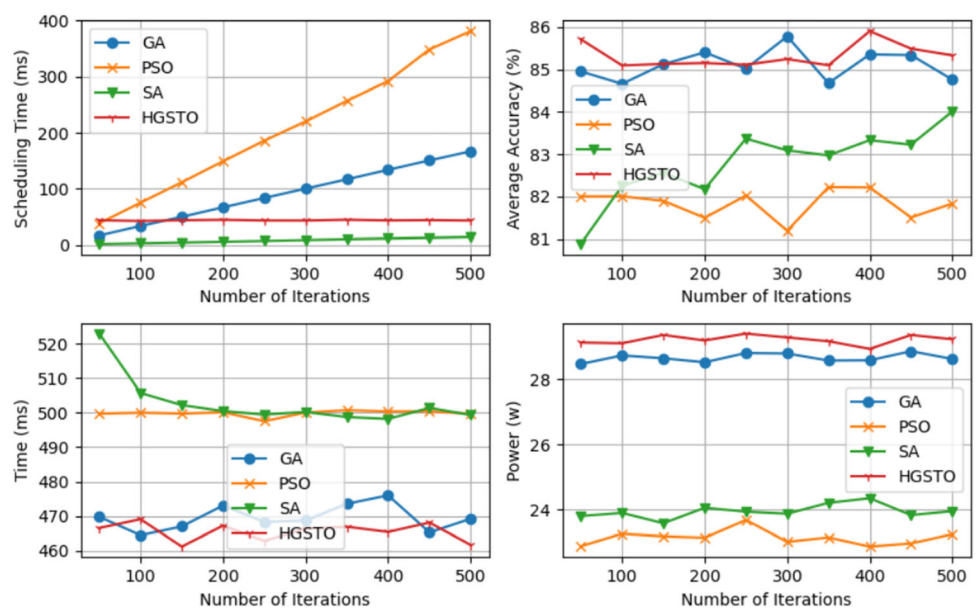


Fig. 7 Performance comparison under varying time constraints with power constraint set to 30 W (Color figure online)

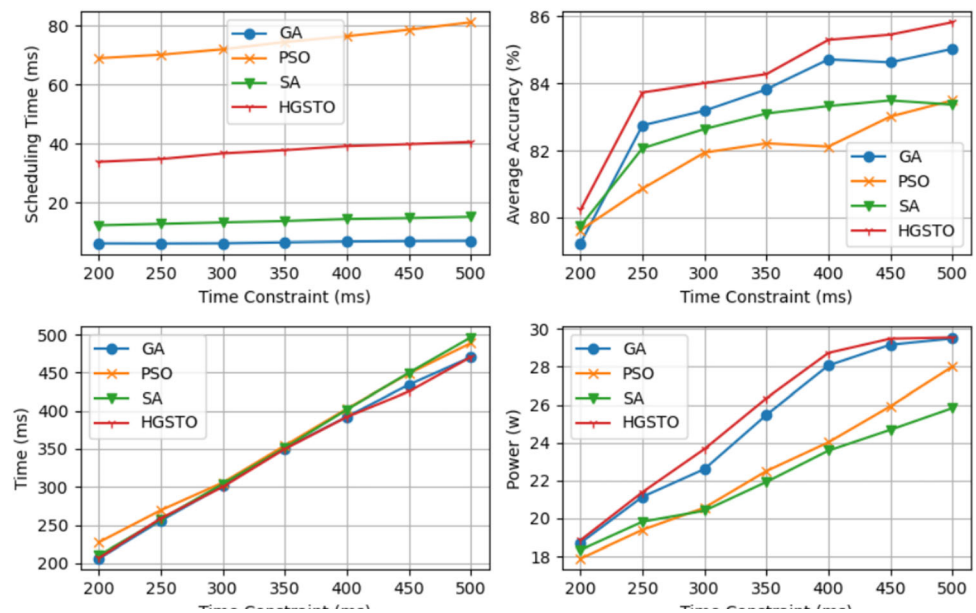
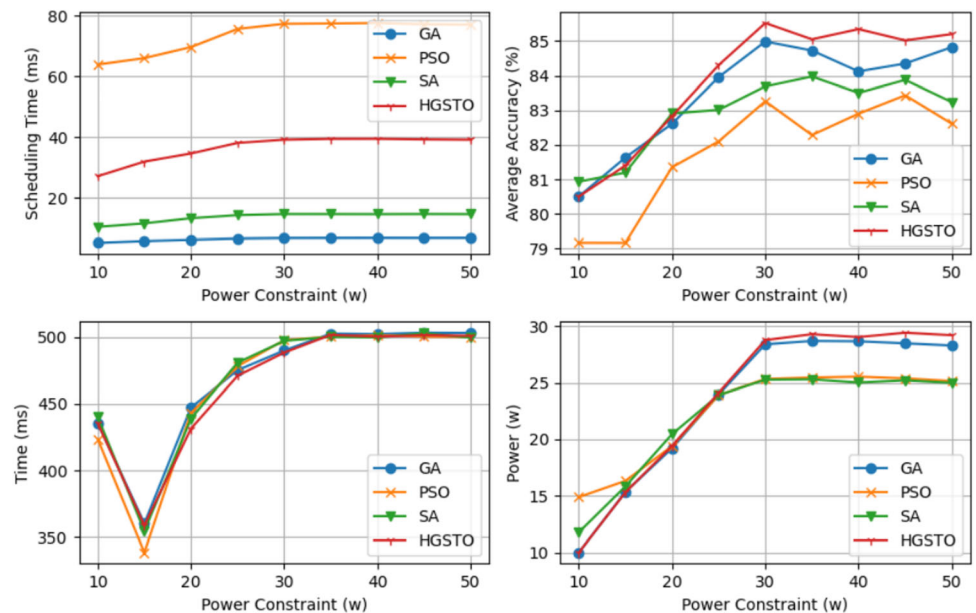


Fig. 8 Performance comparison under varying power constraints with time constraint set to 500 ms (Color figure online)



6 Conclusion

In this work, we investigated the intricate problem of selective inference task offloading in edge computing environments constrained by both time and energy considerations. Specifically, we delved into the decision-making process at the edge devices, where they determine whether to execute inference tasks locally using multiple inference models or offload them to edge servers, aiming to maximize overall accuracy. Demonstrated to be NP-Hard, this problem necessitated the development of an efficient solution approach. To address this challenge, we

introduced HGSTO, a hybrid metaheuristic algorithm that amalgamates the accuracy and versatility of genetic algorithms (GAs) with the stochasticity inherent in simulated annealing (SA). By leveraging the strengths of both techniques, HGSTO harnesses the evolutionary optimization capabilities of GAs while integrating the adaptive search mechanisms of SA, thereby providing a powerful tool for navigating the complex solution space of selective inference task offloading. Through extensive experimentation and evaluation on the ImageNet-Mini dataset, we demonstrated the effectiveness of HGSTO in producing high-quality solutions that optimize both accuracy and resource utilization in edge computing environments.

Fig. 9 Performance comparison under different number of available edge servers with time constraint set to 500 ms and power constraint set to 30 W (Color figure online)

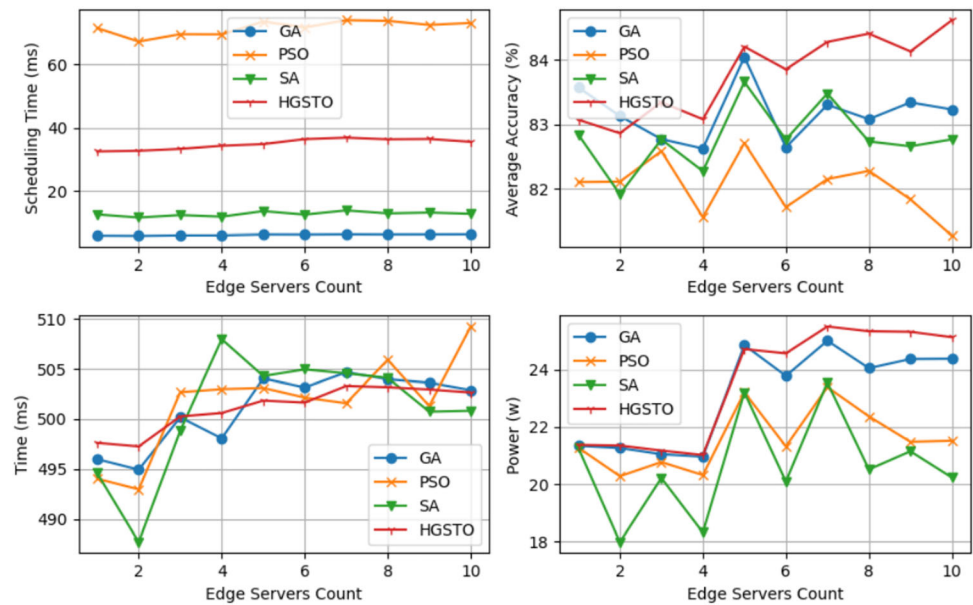


Table 5 Statistical T-test p-value results of HGSTO compared to GA, PSO and SA in terms of fitness and accuracy over 100 test runs

HGSTO against	Fitness p-value	Accuracy p-value
GA	3.96e−36	6.10e−3
PSO	7.13e−150	1.35e−31
SA	3.56e−208	2.87e−16

As a future work, we are considering enhancing HGSTO by integrating deep reinforcement learning (DRL) agents into its framework. By leveraging the capabilities of DRL, HGSTO could potentially benefit from improved decision-making processes and enhanced adaptability to dynamic network environments.

Acknowledgements This article was funded by National Natural Science Foundation of China 61872038.

Author contributions A.B. designed the proposed system. All authors conducted and validated the comparison. All authors wrote the manuscript.

Funding Open Access funding provided by the IReL Consortium.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate

if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A.: Edge computing: a survey. *Future Gener. Comput. Syst.* **97**, 219–235 (2019)
- Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. *IEEE Internet Things J.* **5**(1), 450–465 (2017)
- Lin, H., Zeadally, S., Chen, Z., Labiod, H., Wang, L.: A survey on computation offloading modeling for edge computing. *J. Netw. Comput. Appl.* **169**, 102781 (2020)
- Khelloufi, A., Ning, H., Naouri, A., Sada, A.B., Qamar, A., Khalil, A., Mao, L., Dhelim, S.: A multimodal latent-features-based service recommendation system for the social Internet of Things. *IEEE Trans. Comput. Soc. Syst.* 1–16 (2024). [Online]. Available: <https://ieeexplore.ieee.org/document/10440644/>
- Singh, R., Gill, S.S.: Edge AI: a survey. *Internet Things Cyber-Phys. Syst.* **3**, 71–92 (2023)
- Liu, D., Kong, H., Luo, X., Liu, W., Subramaniam, R.: Bringing AI to edge: from deep learning’s perspective. *Neurocomputing* **485**, 297–320 (2022)
- Dhelim, S., Aung, N., Kechadi, M.T., Ning, H., Chen, L., Lakas, A.: Trust2Vec: large-scale IoT trust management system based on signed network embeddings. *IEEE Internet Things J.* **10**(1), 553–562 (2023). [Online]. Available: <https://ieeexplore.ieee.org/document/9866814/>
- Aung, N., Dhelim, S., Chen, L., Lakas, A., Zhang, W., Ning, H., Chaib, S., Kechadi, M.T.: VeSoNet: traffic-aware content caching for vehicular social networks using deep reinforcement

- learning. *IEEE Trans. Intell. Transp. Syst.* **24**(8), 8638–8649 (2023). [Online]. Available: <https://ieeexplore.ieee.org/document/10070376/>
9. Xiao, H., Xu, C., Ma, Y., Yang, S., Zhong, L., Muntean, G.-M.: Edge intelligence: a computational task offloading scheme for dependent IoT application. *IEEE Trans. Wirel. Commun.* **21**(9), 7222–7237 (2022)
 10. Unhelkar, B., Joshi, S., Sharma, M., Prakash, S., Mani, A.K., Prasad, M.: Enhancing supply chain performance using RFID technology and decision support systems in the industry 4.0—a systematic literature review. *Int. J. Inf. Manag. Data Insights* **2**(2), 100084 (2022)
 11. Kaiwartya, O., Abdullah, A.H., Cao, Y., Lloret, J., Kumar, S., Shah, R.R., Prasad, M., Prakash, S.: Virtualization in wireless sensor networks: fault tolerant embedding for internet of things. *IEEE Internet Things J.* **5**(2), 571–580 (2018)
 12. Nikoloska, I., Zlatanov, N.: Data selection scheme for energy efficient supervised learning at IoT nodes. *IEEE Commun. Lett.* **25**(3), 859–863 (2020)
 13. Fresa, A., Champati, J.P.: Offloading algorithms for maximizing inference accuracy on edge device under a time constraint. *arXiv preprint arXiv:2112.11413* (2021)
 14. Islam, A., Debnath, A., Ghose, M., Chakraborty, S.: A survey on task offloading in multi-access edge computing. *J. Syst. Archit.* **118**, 102225 (2021)
 15. Yang, T., Chai, R., Zhang, L.: Latency optimization-based joint task offloading and scheduling for multi-user MEC system. In: 29th Wireless and Optical Communications Conference (WOCC). *IEEE 2020*, pp. 1–6 (2020)
 16. Liu, C.-F., Bennis, M., Debbah, M., Poor, H.V.: Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Trans. Commun.* **67**(6), 4132–4150 (2019)
 17. Zhang, H., Yang, Y., Huang, X., Fang, C., Zhang, P.: Ultra-low latency multi-task offloading in mobile edge computing. *IEEE Access* **9**, 32569–32581 (2021)
 18. Chen, X., Zhang, J., Lin, B., Chen, Z., Wolter, K., Min, G.: Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments. *IEEE Trans. Parallel Distrib. Syst.* **33**(3), 683–697 (2021)
 19. Li, J., Dai, M., Su, Z.: Energy-aware task offloading in the internet of things. *IEEE Wirel. Commun.* **27**(5), 112–117 (2020)
 20. Xu, Z., Zhao, L., Liang, W., Rana, O.F., Zhou, P., Xia, Q., Xu, W., Wu, G.: Energy-aware inference offloading for DNN-driven applications in mobile edge clouds. *IEEE Trans. Parallel Distrib. Syst.* **32**(4), 799–814 (2020)
 21. Cozzolino, V., Tonetto, L., Mohan, N., Ding, A.Y., Ott, J.: Nimbus: towards latency-energy efficient task offloading for AR services. *IEEE Trans. Cloud Comput.* **11**(2), 1530–1545 (2022). <https://doi.org/10.1109/TCC.2022.3146615>
 22. Abdenacer, N., Abdelkader, N.N., Qammar, A., Shi, F., Ning, H., Dhelim, S.: Task offloading for smart glasses in healthcare: enhancing detection of elevated body temperature. In: 2023 IEEE International Conference on Smart Internet of Things (SmartIoT). *IEEE*, pp. 243–250 (2023)
 23. Younis, A., Tran, T.X., Pompili, D.: Energy-latency-aware task offloading and approximate computing at the mobile edge. In: 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). *IEEE*, pp. 299–307 (2019)
 24. Li, Z., Chang, V., Ge, J., Pan, L., Hu, H., Huang, B.: Energy-aware task offloading with deadline constraint in mobile edge computing. *EURASIP J. Wirel. Commun. Netw.* **2021**, 1–24 (2021)
 25. Tajallifar, M., Ebrahimi, S., Javan, M.R., Mokari, N., Chiaraviglio, L.: Energy-efficient task offloading under e2e latency constraints. *IEEE Trans. Commun.* **70**(3), 1711–1725 (2021)
 26. Liu, K., Peng, J., Li, H., Zhang, X., Liu, W.: Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing. *Future Gener. Comput. Syst.* **64**, 1–14 (2016)
 27. Zhao, M., Yu, J.-J., Li, W.-T., Liu, D., Yao, S., Feng, W., She, C., Quek, T.Q.: Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems. *IEEE Trans. Veh. Technol.* **70**(10), 10925–10940 (2021)
 28. Jiang, H., Dai, X., Xiao, Z., Iyengar, A.: Joint task offloading and resource allocation for energy-constrained mobile edge computing. *IEEE Trans. Mob. Comput.* **22**(7), 4000–4015 (2022)
 29. Mohammad, U., Sorour, S., Hefaida, M.: Task allocation for mobile federated and offloaded learning with energy and delay constraints. In: 2020 IEEE International Conference on Communications Workshops (ICC Workshops). *IEEE*, pp. 1–6 (2020)
 30. Azizi, S., Othman, M., Khamfroush, H.: DECO: a deadline-aware and energy-efficient algorithm for task offloading in mobile edge computing. *IEEE Syst. J.* **17**(1), 952–963 (2022)
 31. Wang, Q., Guo, S., Liu, J., Yang, Y.: Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing. *Sustain. Comput. Inform. Syst.* **21**, 154–164 (2019)
 32. Alameddine, H.A., Sharafeddine, S., Sebbah, S., Ayoubi, S., Assi, C.: Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. *IEEE J. Sel. Areas Commun.* **37**(3), 668–682 (2019)
 33. Ni, W., Tian, H., Lyu, X., Fan, S.: Service-dependent task offloading for multiuser mobile edge computing system. *Electron. Lett.* **55**(15), 839–841 (2019)
 34. Alfakih, T., Hassan, M.M., Gumaie, A., Savaglio, C., Fortino, G.: Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. *IEEE Access* **8**, 54074–54084 (2020)
 35. Huang, L., Feng, X., Zhang, C., Qian, L., Wu, Y.: Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digit. Commun. Netw.* **5**(1), 10–17 (2019)
 36. Li, Z., Zhu, Q.: Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing. *Information* **11**(2), 83 (2020)
 37. Abbas, A., Raza, A., Aadil, F., Maqsood, M.: Meta-heuristic-based offloading task optimization in mobile edge computing. *Int. J. Distrib. Sens. Netw.* **17**(6), 15501477211023020 (2021). <https://doi.org/10.1177/15501477211023021>
 38. Ezugwu, A.E., Pillay, V., Hirasen, D., Sivanarain, K., Govender, M.: A comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: some initial results. *IEEE Access* **7**, 43979–44001 (2019)
 39. Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A.K.: Chapter 10—metaheuristic algorithms: a comprehensive review. In: Sangaiah, A.K., Sheng, M., Zhang, Z. (eds.) *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, ser. *Intelligent Data-Centric Systems*, pp. 185–231. Academic Press, New York (2018)
 40. Knuth, D.E.: *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2. Addison-Wesley Professional, Boston (2014)
 41. Alhijawi, B., Awajan, A.: Genetic algorithms: theory, genetic operators, solutions, and applications. *Evol. Intell.* **17**(3), 1245–1256 (2023)
 42. Delahaye, D., Chaimatanan, S., Mongeau, M.: Simulated annealing: from basics to applications. In: Gendreau, M., Potvin, J.-Y. (eds.) *Handbook of Metaheuristics*, pp. 1–35. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_1
 43. Haddar, B., Khemakhem, M., Hanafi, S., Wilbaut, C.: A hybrid quantum particle swarm optimization for the multidimensional knapsack problem. *Eng. Appl. Artif. Intell.* **55**, 1–13 (2016)

44. Bansal, J.C., Deep, K.: A modified binary particle swarm optimization for knapsack problems. *Appl. Math. Comput.* **218**(22), 11042–11061 (2012)
45. Tanha, M., Hosseini Shirvani, M., Rahmani, A.M.: A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. *Neural Comput. Appl.* **33**, 16951–16984 (2021)
46. Fanian, F., Bardsiri, V.K., Shokouhifar, M.: A new task scheduling algorithm using firefly and simulated annealing algorithms in cloud computing. *Int. J. Adv. Comput. Sci. Appl.* **9**(2), 195–202 (2018)
47. Chen, Y., Hao, J.-K.: Memetic search for the generalized quadratic multiple knapsack problem. *IEEE Trans. Evol. Comput.* **20**(6), 908–923 (2016)
48. Kierkosz, I., Luczak, M.: A hybrid evolutionary algorithm for the two-dimensional packing problem. *CEJOR* **22**, 729–753 (2014)
49. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc. (2016)
50. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015)
51. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet v2: practical guidelines for efficient CNN architecture design. In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2018)
52. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.




Abdelkarim Ben Sada Received his BSc in Computer Science in 2014 from the University of Djelfa Algeria, and his MSc degree in 2016 majoring in Networking and Distributed Systems from the University of Laghouat Algeria. He is currently pursuing his PhD degree at the University of Science and Technology Beijing China. His research interests include Computer Vision, Machine Learning and Internet of Things.



Amar Khelloufi Received the B.S. degree (Hons.) in computer science from the Faculty of Sciences and Technology, Ziane Achour University of Djelfa, Djelfa, Algeria, in 2012, and the M.S. degree in distributed information systems from the Faculty of Sciences, University of Boumerdès, Boumerdès, Algeria, in 2014. He is currently pursuing the Ph.D. degree with the School of Computer and Communication Engineering, University of Science and

Technology Beijing, Beijing, China. His current research focuses on Internet of Things, blockchain applications, edge computing, and distributed systems.



Abdenacer Naouri He is currently a Ph.D. candidate at the University of Science and Technology Beijing China, Beijing, China. He received his B.S. degree in computer science from the University of Djelfa Algeria, in 2011, and the M.Sc. degree in networking and distributed systems from the University of Laghouat Algeria, Laghouat, Algeria, in 2016. His current research interests include Cloud computing, Smart communication, machine learning, Internet of vehicles and Internet of Things.



Huansheng Ning Received his B.S. degree from Anhui University in 1996 and his Ph.D. degree from Beihang University in 2001. Now, he is a professor and vice dean of the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His current research focuses on the Internet of Things and general cyberspace. He has presided many research projects including Natural Science Foundation of China, National High Technology Research and Development Program of China (863 Project). He has published more than 150 journal/conference papers, and authored 5 books. He serves as an associate editor of *IEEE Systems Journal* (2013–Now), *IEEE Internet of Things Journal* (2014–2018), and as steering committee member of *IEEE Internet of Things Journal* (2016–Now).