2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3315-1723-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDCS63083.2025.00108

TSAJS: Efficient Multi-Server Joint Task Scheduling Scheme for Mobile Edge Computing

Chaoqun Li[†], Rongsheng Fan[†], Hesong Wang[†], Mingda Han[†], Si Wu[†], Feng Li[†], Pengfei Hu[†*]
[†] School of Computer Science and Technology, Shandong University, Qingdao, China
{chaoqunli, fanrongsheng, wanghesong, mingdhan}@mail.sdu.edu.cn, {kobesi, fli, phu}@sdu.edu.cn

***Abstract*—Mobile Edge Computing (MEC) utilizes edge servers to offload the computational burden from cloud infrastructure. By providing low-latency and high-bandwidth services, MEC enables mobile users and IoT devices to efficiently offload and execute computational tasks at the network edge. However, optimizing communication and computational resources in a multi-user, multi-server MEC environment remains a significant challenge. In this paper, we propose TSAJS, an efficient multi-server joint task scheduling scheme designed to enhance the effectiveness of MEC offloading. We model the task offloading and resource allocation problem as a Mixed-Integer Nonlinear Programming (MINLP) problem, aiming to maximize user offloading gain by minimizing task completion time and energy consumption. A heuristic algorithm for offloading is introduced by combining threshold-triggering and simulated annealing to effectively avoid local optima and converge toward the global optimum. Meanwhile, the optimal solution for resource allocation is derived using the Karush-Kuhn-Tucker (KKT) conditions. Experimental results demonstrate that TSAJS delivers near-optimal performance, outperforming traditional methods in terms of user offloading effectiveness. Its efficiency enables solution finding within polynomial time, while also adapting to the preferences of users and service providers.**

***Index Terms*—Mobile edge computing, Internet of Things, distributed computing system, computation offloading, multi-server resource allocation, approximate optimization.**

I. Introduction

With the rapid proliferation of connected devices, the upper layers of the internet are experiencing unprecedented pressure. Traditionally, many critical applications and services have relied on centralized cloud architectures. However, this model is increasingly revealing significant bottlenecks, including network congestion and latency issues, which hinder scalability and performance in real-time applications [1], [2]. In response to these challenges, Edge Computing (EC) has emerged as an innovative computing paradigm that addresses the limitations of traditional cloud computing. By deploying computing resources closer to users and data sources at the network edge, EC reduces data transmission distances, lowers latency, and enhances bandwidth utilization, thereby improving service efficiency and user experience [3]. From instant responses in smart homes to safe navigation in autonomous driving, EC, with its unique advantages, brings unprecedented convenience and efficiency to various industries [4].

As a prominent branch of EC, Mobile Edge Computing (MEC) further amplifies the advantages of this technology. By deploying servers at the edge of the network, MEC provides powerful computational support for mobile devices, enabling seamless offloading of complex tasks. This not only alleviates the processing burden on devices but also extends battery life and improves overall performance [5]. MEC has proven instrumental in various real-world applications, including traffic management in smart cities, real-time monitoring in industrial production, interactive classrooms in remote education, and immersive virtual reality experiences [6]. However, as the scope of application scenarios continues to expand, achieving efficient task offloading and rational resource allocation in complex environments with multiple users and servers has become a significant challenge [7], [8]. The decision-making process for task offloading is inherently complex and dynamic, influenced by factors such as task characteristics, network conditions, and server loads. Addressing these complexities has become a critical research focus, as well as a key obstacle, in advancing the field of MEC [9].

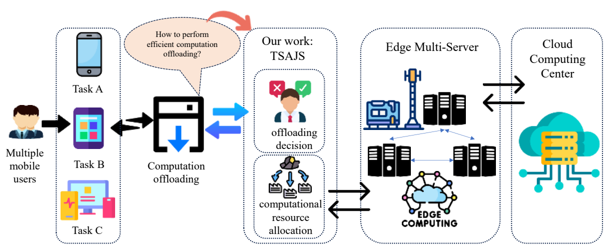


Fig. 1. Efficient task offloading and computing resource allocation for multiple mobile users through TSAJS in a multi-server MEC network.

The communication requirements between devices and MEC servers in the uplink wireless channel inevitably introduce additional latency and energy consumption costs. In environments with numerous offloading users, the limited computational resources of MEC servers become a significant bottleneck, constraining task execution efficiency. Addressing this challenge requires making intelligent offloading decisions and efficiently allocating execution resources, which have become central topics for achieving effective computational offloading. Previous research has made some progress in this field, focusing on optimizing offloading strategies [6], communication resources [10], [11], or computational resources [12] to partially mitigate this challenge. Recently, Kuang et

* Pengfei Hu is the corresponding author.

al. [13] explored partial offloading scheduling and power allocation strategies in mobile edge computing systems. Liu et al. [14] optimized the delay of computational offloading using NOMA-OMA hybrid transmission technology. Li et al. [15] achieved energy-efficient task offloading through dynamic voltage scaling technology. Although these studies provide valuable insights into task offloading in a single-server MEC, they fall short of addressing the complexity of real-world scenarios. In practical applications, multi-server, multi-user scenarios are increasingly common, necessitating a reevaluation and optimization of task offloading strategies.

As shown in Figure 1, our approach enhances the traditional MEC system by designing a strategic framework for multi-server MEC-assisted networks. This framework optimizes user computation offloading in ultra-dense multi-cell environments, where each base station (BS) is equipped with an MEC server. The multi-server setup ensures balanced resource allocation, effectively addressing overload issues, and improving load distribution [16]. Meanwhile, it reduces user interference and resource contention, boosting processing efficiency in multi-task scenarios. To achieve this, a comprehensive understanding of user tasks, MEC resources, and wireless channel conditions is essential. This is enabled through Coordinated Multi-Point (CoMP) systems for channel information sharing or Cloud Radio Access Networks (C-RAN), where all BSs connect to a unified Baseband Unit (BBU). The centralized access to system state provided by C-RAN enhances coordination and resource management. While this approach introduces increased backhaul traffic and signaling requirements, the tight integration of BSs in C-RAN mitigates these challenges, offering a more efficient and scalable solution [17]. We summarize our main contributions as follows:

- We propose TSAJS, an efficient multi-server joint task scheduling scheme that models the communication and computing resource optimization in a multi-user, multi-server MEC system as a MINLP problem, maximizing task offloading gains through optimized offloading and resource allocation.
- We develop a Threshold-Triggered Simulated Annealing (TTSA) algorithm for task offloading, which enhances search efficiency and avoids local optima. Based on the offloading decisions made by the TTSA algorithm, the Karush-Kuhn-Tucker (KKT) conditions are then applied to optimize the computing resource allocation, ensuring both efficiency and feasibility.
- TSAJS can find near-optimal solutions in polynomial time, adapting to user and service provider preferences. Simulation results demonstrate its superiority in offloading efficiency, outperforming existing solutions.

II. Related Works

Mobile Edge Computing (MEC), an edge computing architecture standard introduced by Professor Satyanarayanan, focuses on deploying computational and storage resources at the network edge. Designed to address the unique constraints of mobile networks, MEC has been recognized across three key industries for its specialized applications. Originally proposed by Nokia Siemens Networks in 2013, MEC was designed to operate within the Radio Access Cloud Server (RACS) and fully integrate with multimodal base stations, such as Flexi [18]. Building upon this foundation, the European Telecommunications Standards Institute (ETSI) introduced several related variants to enhance MEC's capabilities [19]. Today, MEC has become a cornerstone for delivering diverse computing applications and content services, driven by service providers leveraging edge network infrastructure. To further regulate and standardize its application in the Radio Access Network (RAN), an industry standards group has been established, highlighting its growing importance in modern network ecosystems [20].

The MEC network comprises multiple edge servers deployed at base stations, each functioning as a compact data center tailored to meet specific service requirements. In the context of IoT and 5G, various customized solutions for MEC have been proposed. The ITU's M.2083 report highlights three key 5G application scenarios: enhanced virtual reality [21], IoT [22], and vehicular networks [23], along with use cases such as video streaming analysis and acceleration [24], enabling faster response times and improved quality of service (QoS). As the number of connected devices continues to grow, the sheer volume of generated data places significant strain on centralized cloud architectures, leading to communication bottlenecks and inefficiencies [25]. Addressing these challenges requires advancing MEC research across five critical aspects: delay-sensitive applications [26], real-time processing and edge intelligence, economic considerations [27], security and privacy requirements, and enabling technologies for MEC. At the same time, MEC must offer effective solutions to mitigate the challenges arising from its implementation.

Recent research on MEC has emphasized its exceptional capabilities in computation offloading, which involves efficiently allocating users' compute-intensive tasks—whether independent tasks or workflows—to edge servers with sufficient computing resources [28]. The processed results are then returned to the mobile devices. Task offloading encompasses several types, including partial task offloading, full task offloading, code offloading, and horizontal offloading. For full task offloading, methods have been developed to minimize both the total resource cost and the latency of user requests [29]. In the case of partial task offloading, a distributed approach has been proposed to manage offloading tasks, allowing each task to be arbitrarily partitioned at the bit level for optimized partial offloading [30].

An examination of the literature on computational offloading tasks reveals that current research objectives can be categorized into four primary evaluation criteria: minimizing latency, reducing energy consumption, maximizing profit, and enhancing system utility. To address these objectives, a wide range of mathematical and computational methods has been utilized, including Mixed-Integer Nonlinear Programming (MINLP) [31], Successive Convex Approximation (SCA) [32], and sub-optimal algorithms based on hierarchical genetic algorithms and particle swarm optimization [33]. Furthermore, advanced methodologies such as game theory, Lyapunov optimalization [34], Markov Decision Processes (MDP) or Deep Reinforcement Learning (DRL) [35] [36], and adaptive learning techniques leveraging Multi-Armed Bandit (MAB) theory have been proposed to tackle the inherent complexities of computational offloading. In scenarios involving static computation offloading, where multiple users concurrently transmit their requests to a MEC network, the challenge can frequently be represented as a MINLP. While global optimal solutions can typically be obtained using conventional methods, these approaches often come with high computational complexity. Therefore, in scenarios where latency requirements are more relaxed, alternative methods with lower computational overhead are often employed to solve the computation offloading problem.

Our work distinguishes itself from the aforementioned literature in several key aspects. It models the scenario under the assumption of a fixed data rate for wireless links, incorporating both task allocation and resource allocation while considering variations in wireless communication channels and multi-user interference. The study comprehensively addresses the optimization of energy consumption and task execution time through the design of a parameter-tunable objective function. To achieve optimal efficiency under these conditions, an improved simulated annealing algorithm is utilized. In addition, this work pays special attention to offloading efficiency and adopts a comprehensive approach to calculate offloading decisions and wireless resource allocation in multi-user and multi server systems.

III. Preliminary

A. System Model

In a multi-user, multi-server MEC system, each base station (BS) integrates an MEC server to serve resource-limited mobile users (e.g., smartphones, wearable devices). These servers, whether physical or virtual, facilitate wireless task offloading for connected devices. In this paper, we represent the sets of mobile users and MEC servers as $\mathcal{U} = \{1, 2, \ldots, U\}$ and $\mathcal{S} = \{1, 2, \ldots, S\}$, respectively. For model design, we refer to concepts from [37]. To streamline our discussion, we use "MEC server" and "BS" interchangeably. Subsequently, we elaborate on the models for user computing tasks, MEC computing resources, and offloading utility. The key notations used in this paper have been summarized in Table I.

1) User Computation Tasks: We presume that each individual user u within the framework possesses a singular, non-divisible computational assignment, denoted as T_u. This assignment is atomic, meaning it cannot be subdivided into smaller components. Each computational task T_u is characterized by a pair of parameters, specifically $\langle d_u, w_u \rangle$. In this context, d_u[bits] signifies the volume of data necessary to relocate the program's execution (encompassing system settings, program instructions, and input values) from the user's personal device to the Multi-Access Edge Computing server. Meanwhile, w_u[cycles] embodies the task's computational load, or the amount of processing power needed to accomplish it. The precise magnitudes of d_u and w_u can be ascertained through thorough task execution analysis [38]. Users have the option to execute these tasks locally on their devices or delegate them to the MEC server for completion. While offloading tasks to the MEC server can lead to energy savings on the user's device, it also entails additional time and energy expenditure for transmitting the task inputs via the uplink.

TABLE I

Overview of Key Symbols and Their Meanings

Symbol	Meaning
$\mathcal{U}$	Collection of U individual users
$\mathcal{S}$	Assembly of S base stations/MEC servers
T_u	User u's computational assignment
d_u	Data input for the computational task T_u
c_u	Computational burden of task T_u
f_u^{local}	User u's local processing power
E_u^{local}	Energy expended by user u for local task execution
E_u	Energy used by user u during task offloading
κ	Energy efficiency factor of user's device
t_u^{local}	Duration for local execution of task T_u
t_u^{transfer}	Time taken to transmit task T_u to the MEC server
t_u^{process}	Time required to process task T_u at the MEC server
t_u	Total latency for user u when offloading a task
B	Bandwidth allocated for uplink communication
$\mathcal{N}$	Group of N orthogonal sub-channels
x_{us}^j	Indicator for task offloading, for all $u \in \mathcal{U}$, $s \in \mathcal{S}$, $j \in \mathcal{N}$
$\mathcal{G}$	Set of possible task offloading configurations
$\mathcal{X}$	Strategy for task offloading
$\mathcal{U}_{\text{off}}$	Collection of users opting for task offloading
$\mathcal{U}_s$	Users assigning tasks to server s
h_{us}^j	Signal strength for user u to BS s on sub-channel j
p_u	Transmission strength of user u
γ_{us}^j	SINR for user u to BS s on sub-channel j
R_{us}	Data transfer rate from user u to BS s
f_s	Processing capacity of server s
f_{us}	CR assigned by server s to user u's task
$\mathcal{F}$	Allocation plan for computational resources
J_u	Benefit of offloading for user u
β_u^{time}	User u's priority on task completion speed
β_u^{energy}	User u's emphasis on energy conservation
λ_u	Service provider's preference for user u

Let f_u^{local} (in CPU cycles/second) denote user u's local computing power. If u executes task T_u locally, the completion time is $t_u^{local} = \frac{w_u}{f_u^{local}}$ [seconds]. For local task energy consumption, we use the model $\varepsilon = \kappa f^2$, where κ is a chip-dependent factor and f is the CPU frequency. Thus, user u's energy consumption E_u^{local} [J] for task T_u is:

$$E_u^{local} = \kappa \left(f_u^{local}\right)^2 w_u \tag{1}$$

2) Task Uploading: In the scenario where users migrate their computing tasks to MEC servers, the overall latency consists of three main parts. First, it is the time required for users to send task data to the MEC server through the uplink, denoted as t_{upload}^u [seconds]. The second is the time required by the MEC server to process the task, denoted as t_{execute}^u[seconds]. Finally, it is the time when the server returns the processing result to the user through the downlink. Generally, we ignore the transmission delay in the downlink in our model due to the small amount of output data and the fast data transmission rate in the downlink. However, if the downlink latency becomes significant, our algorithm can still adapt by taking into account the actual downlink rate and the output data size.

This study focuses on an uplink system employing OFDMA (orthogonal frequency division multiple access) technology, where the total frequency band B is divided into N sub-bands of equal width, each with a width of $W = \frac{B}{N}$[Hz] [39]. To ensure that transmissions between different users connected to the same base station do not interfere with each other, each user will be assigned a unique sub-band. In this framework, each base station possesses the capability to theoretically handle N users concurrently. We designate the set of sub-bands allocated to each base station as $\mathcal{N} = \{1, \ldots, N\}$. The migration status of a task is indicated by the binary variable x_{us}^j, where $u \in \mathcal{U}$ (the set of users), $s \in \mathcal{S}$ (the set of base stations), and $j \in \mathcal{N}$. Specifically, $x_{us}^j = 1$ signifies that user u's task T_u has been migrated to base station s on sub-band j, while $x_{us}^j = 0$ indicates otherwise. The collection of all such migration variables is denoted as $\mathcal{G} = \{x_{us}^j | u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}\}$. The set of actual task migrations, referred to as the offloading policy $\mathcal{X}$, is then defined as $\mathcal{X} = \{x_{us}^j \in \mathcal{G} | x_{us}^j = 1\}$. Given that each task can either be processed locally or offloaded to a single MEC server, a valid offloading policy must adhere to the following constraint:

$$\sum_{s \in S} \sum_{j \in N} x_{us}^j \leq 1, \quad \forall u \in \mathcal{U}. \tag{2}$$

Furthermore, we introduce the notation $\mathcal{U}_s = \{u \in \mathcal{U} | \sum_{j \in \mathcal{N}} x_{us}^j = 1\}$ to represent the subset of users who have offloaded their tasks to server s. Additionally, $\mathcal{U}_{\text{offload}} = \bigcup_{s \in \mathcal{S}} \mathcal{U}_s$ denotes the set of all users who have chosen to offload their tasks.

In the uplink scenario, we consider a setup where each user and base station utilize a single antenna for transmission, leaving the exploration of multi-antenna configurations at the base station for future work. We define h_{jus} as the channel gain between user u and base station s on sub-band j, encompassing factors like path loss, shadowing, and antenna gain. The user-base station association process occurs over a long-term scale, significantly exceeding the rapid signal fading timescale. Consequently, we assume that the impact of fast fading can be averaged out during this association process [40]. For users in the set $\mathcal{U}_{\text{offload}}$, let p_u represent the constant transmit power of user u when uploading task data d_u to the base station. Note that for users not offloading, i.e., $u \notin \mathcal{U}_{\text{offload}}$, $p_u = 0$. Due to the orthogonal transmission of users across different sub-bands, intra-cell interference in the uplink is minimized. However, users may still experience interference from adjacent cells. The Signal-to-Interference-plus-Noise Ratio (SINR) for user u transmitting to base station s on sub-band j is given by:

$$\gamma_{us}^j = \frac{p_u h_{us}^j}{\sum_{r \in \mathcal{S}, r \neq s} \sum_{k \in \mathcal{U}_r} x_{kr}^j p_k h_{ks}^j + \sigma^2}, \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N} \tag{3}$$

where σ^2 is the background noise variance, and the double summation in the denominator accounts for the interference from all users associated with other base stations on the same sub-band j.

Since each user transmits on only one sub-band, the achievable rate [bits/s] for user u when sending data to base station s is expressed as:

$$R_{us}(\mathcal{X}) = W \log_2 \left(1 + \sum_{j \in \mathcal{N}} \gamma_{us}^j\right) \tag{4}$$

where $\gamma_{us} = \sum_{j \in \mathcal{N}} \gamma_{us}^j$ is the aggregate SINR across all sub-bands. Additionally, we define $x_{us} = \sum_{j \in \mathcal{N}} x_{us}^j$ for all $u \in \mathcal{U}$ and $s \in \mathcal{S}$. The transmission time for user u to upload its task data d_u in the uplink is then calculated as:

$$t_{\text{upload}}^u = \sum_{s \in \mathcal{S}} \frac{x_{us} d_u}{R_{us}(\mathcal{X})}, \quad \forall u \in \mathcal{U}. \tag{5}$$

3) MEC Computing Resources: Each MEC server situated at the base stations within the system possesses the capability to offer computation offloading services to numerous users concurrently. The computational resources made available by each MEC server to its associated users are quantified by a computation rate, denoted as f_s, which represents the number of CPU cycles per second. Upon receiving an offloading task from a user, the MEC server undertakes the execution of that task and subsequently returns the results to the user once the execution is finished. The policy for allocating computational resources within the system is formulated as $\mathcal{F} = \{f_{us} | u \in \mathcal{U}, s \in \mathcal{S}\}$, where f_{us} [cycles/s] > 0 signifies the allocation of computational resources by base station s for a task T_u offloaded by user u. Hence, it follows that $f_{us} = 0$ for any user u not associated with base station s. Furthermore, a viable computational resource allocation policy must adhere to the constraint of computational resources, which is articulated as:

$$\sum_{u \in \mathcal{U}} f_{us} \leq f_s, \quad \forall s \in \mathcal{S}. \tag{6}$$

For a given allocation of computational resources $\{f_{us}, s \in \mathcal{S}\}$, the time required to execute task T_u on the MEC server is calculated as:

$$t_{\text{execute}}^u = \sum_{s \in \mathcal{S}} \frac{x_{us} w_u}{f_{us}}, \quad \forall u \in \mathcal{U}. \tag{7}$$

4) User Offloading Utility: In the context of the offloading strategy $\mathcal{X}$, alongside the designated transmission power p_u and the allocated computation resources f_{us}, the overall delay encountered by user u during task offloading is determined by the formula:

$$t_u = t_{\text{upload}}^u + t_{\text{execute}}^u = \sum_{s \in \mathcal{S}} x_{us} \left(\frac{d_u}{R_{us}(\mathcal{X})} + \frac{w_u}{f_{us}}\right), \quad \forall u \in \mathcal{U}. \tag{8}$$

The energy expenditure E_u [J] of user u associated with the upload transmission, taking into account the power amplifier efficiency ξ_u, which we normalize to 1 for simplicity ($\xi_u = 1, \forall u \in \mathcal{U}$), is computed as:

$$E_u = p_u t^u_{\text{upload}} = p_u d_u \sum_{s\in\mathcal{S}} \frac{x_{us}}{R_{us}(\mathcal{X})}, \quad \forall u \in \mathcal{U}. \tag{9}$$

Within mobile cloud computing frameworks, user QoE (Quality of Experience) is primarily gauged by task completion time and energy usage. For isolated tasks, we evaluate the relative enhancements in task completion time and energy consumption through the ratios $\frac{t_u^{local}-t_u}{t_u^{local}}$ and $\frac{E_u^{local}-E_u}{E_u^{local}}$, respectively. Consequently, the offloading benefit for user u is formulated as:

$$J_u = \left(\beta_u^{time}\frac{t_u^{local}-t_u}{t_u^{local}} + \beta_u^{energy}\frac{E_u^{local}-E_u}{E_u^{local}}\right)\sum_{s\in\mathcal{S}} x_{us} \tag{10}$$

where β_u^{time} and β_u^{energy} denote the respective weights reflecting user u's preferences for time and energy savings. Within the framework where $\beta_u^{time}, \beta_u^{energy} \in [0,1]$ and $\beta_u^{time} + \beta_u^{energy} = 1$ for all users $u \in \mathcal{U}$, these parameters serve as indicators of a user's priorities concerning task completion speed and energy usage. Specifically, a user with a low battery might choose to increase β_u^{energy} while decreasing β_u^{time}, thereby prioritizing energy preservation over rapid task execution.

Offloading tasks to MEC servers can cause delays and degrade user experience if resources are limited. Therefore, users should only offload if the benefit J_u is positive. A joint optimization approach, considering offloading scheduling, wireless resource allocation, and computational management, is essential to maximize benefits and minimize negative impacts on Quality of Experience (QoE).

B. Problem Formulation

In this section, we articulate the challenge of integrating task offloading with resource distribution, and subsequently provide an outline of our methodology for breaking down and addressing this complex problem.

1) Joint Task Offloading and Resource Allocation Problem: In the realm of Joint Task Offloading and Resource Allocation (JTORA), our primary goal is to enhance the overall system utility, which we define as the weighted aggregate of individual user offloading utilities. It's important to note that, in this context, we've kept the user transmit power constant, denoted as p_u, as discussed in earlier sections. Given a set of offloading decisions $\mathcal{X}$ and computational resource allocations $\mathcal{F}$, the weighted sum of user offloading utilities can be mathematically represented as:

$$J(\mathcal{X},\mathcal{F}) = \sum_{u\in\mathcal{U}} \lambda_u J_u, \tag{11}$$

where J_u is derived from equation (10), and $\lambda_u \in (0,1]$ signifies the resource provider's preference for user u, applying to all $u \in \mathcal{U}$. This preference factor λ_u can be dynamically adjusted based on user payments, with higher-paying users receiving greater priority for offloading. Furthermore, λ_u may also be influenced by user characteristics and the urgency of their computational tasks.

For example, in emergency situations involving public safety personnel, such as police officers or first responders using mobile devices, it's crucial to assign these users a higher λ_u value to ensure their tasks are given top priority. Consequently, the JTORA problem is formulated as an optimization challenge to maximize this utility function, while adhering to the following constraints:

$$\max_{\mathcal{X},\mathcal{F}} J(\mathcal{X},\mathcal{F}), \tag{12a}$$

$$\text{s.t.} \quad x_{us}^j \in \{0,1\}, \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}, \tag{12b}$$

$$\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}} x_{us}^j \le 1, \forall u \in \mathcal{U}, \tag{12c}$$

$$\sum_{u\in\mathcal{U}} x_{us}^j \le 1, \forall s \in \mathcal{S}, j \in \mathcal{N}, \tag{12d}$$

$$f_{us} > 0, \forall u \in \mathcal{U}_s, s \in \mathcal{S}, \tag{12e}$$

$$\sum_{u\in\mathcal{U}} f_{us} \le f_s, \forall s \in \mathcal{S}. \tag{12f}$$

In the context of our mathematical model, the constraints serve vital practical purposes: Constraint (12b) ensures that each computing task is processed either locally or offloaded to a single sub-band of a single server, thus enforcing the uniqueness of task offloading. This prevents any task from being redundantly processed or from being allocated to multiple resources simultaneously. Constraint (12c) stipulates that each user can only select one service on any sub-band for task offloading. This avoids over-allocation of resources and potential conflicts, ensuring that resources are utilized efficiently and without contention. Constraint (12d) limits each base station to serving at most one user on the same sub-band. This optimization of network resource allocation helps prevent service congestion and ensures that network resources are distributed fairly among users. Constraints (12e) and (12f) relate to the allocation of computing resources. They require each MEC server to allocate a specific amount of computing resources to each user it serves, with the total allocated resources not exceeding the server's capacity. This ensures that resource allocation is both rational and feasible, preventing any server from being overloaded.

The JTORA problem, as formulated in Eq.(12), falls under the category of Mixed Integer Nonlinear Programming (MINLP). Solving this problem optimally often involves dealing with exponential time complexity, which can be computationally prohibitive. Given that the number of variables increases linearly with the number of users, MEC servers, and sub-bands, our objective is to develop a solution that balances computational efficiency with near-optimal performance.By carefully designing our algorithm, we aim to achieve a scheme that not only offers competitive performance but is also practical to implement and deploy in real-world systems. This approach significantly reduces computational costs and improves efficiency, all while maintaining system performance at a high level.

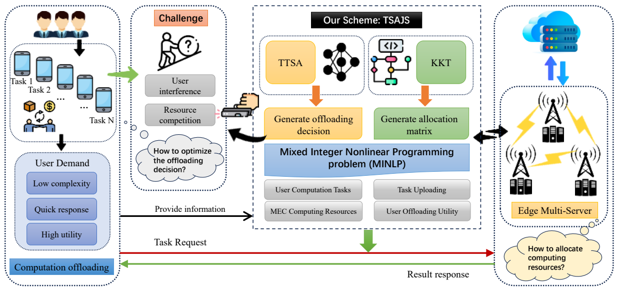


Fig. 2. TSAJS Method technical framework.

2) Problem Decomposition: Through an in-depth examination of the objective function and constraints inherent in the JTORA problem stated in (12), we have identified a pivotal optimization technique. By momentarily considering the binary variable x_{us} as fixed, the original problem can be adeptly decomposed into several manageable sub-problems. These sub-problems exhibit independent objective functions and relatively straightforward constraints. Leveraging this characteristic, we employ the Tamer decomposition approach to convert the intricate original problem into an equivalent master problem accompanied by a series of less complex sub-problems [41]. Initially, we reframe the JTORA problem as outlined in Eq. (12):

$$\max(\max_{\mathcal{X},\mathcal{F}} J(\mathcal{X},\mathcal{F})), \tag{13a}$$

$$\text{subject to} \quad (12b), (12c), (12d), (12e), (12f) \tag{13b}$$

It is noteworthy that the constraints imposed on the task offloading decision variable $\mathcal{X}$ in equations (12b), (12c), and (12d) are decoupled from those governing the resource allocation strategies $\mathcal{F}$ in equations (12e) and (12f). Therefore, tackling the problem in equation (13) is fundamentally equivalent to resolving the ensuing Task Offloading (TO) optimization challenge:

$$\max_{\mathcal{X}} \hat{J}(\mathcal{X}), \tag{14a}$$

$$\text{subject to} \quad (12b), (12c), (12d) \tag{14b}$$

Here, $\hat{J}(\mathcal{X})$ represents the optimal value function associated with the Resource Allocation (RA) problem, which is formulated as:

$$\hat{J}(\mathcal{X}) = \max_{\mathcal{F}} J(\mathcal{X},\mathcal{F}), \tag{15a}$$

$$\text{subject to} \quad (12e), (12f) \tag{15b}$$

The decomposition of problem (12) into subproblems (14) and (15) guarantees the maintenance of optimal solutions. In the subsequent section, we will elaborate on our approaches to tackling the Resource Allocation (RA) and Task Offloading (TO) challenges individually, aiming to arrive at a comprehensive solution for the initial integrated task offloading and resource allocation issue.

IV. The TSAJS Method

We now introduce our TSAJS method for tackling the JTORA problem. Our approach involves first resolving the RA problem outlined in (15) and subsequently utilizing its solution to address the TO problem presented in (14).

To begin, we take into account a viable task offloading decision $\mathcal{X}$ that satisfies the constraints (12b), (12c), and (12d). By incorporating the expression for J_u from equation (10), we can restate the objective function in equation (15a) as follows:

$$J(\mathcal{X},\mathcal{F}) = \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\lambda_u\left(\beta_u^{time}+\beta_u^{energy}\right) - V(\mathcal{X},\mathcal{F}), \tag{16}$$

where $V(\mathcal{X},\mathcal{F})$ is defined as:

$$V(\mathcal{X},\mathcal{F}) = \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\lambda_u\left(\frac{\beta_u^{ttime}t_u}{t_u^{local}}+\frac{\beta_u^{energy}E_u}{E_u^{local}}\right). \tag{17}$$

It's evident that the initial term on the right-hand side of equation (16) remains constant for a given offloading decision, while $V(\mathcal{X},\mathcal{F})$ signifies the cumulative offloading cost for all users opting for offloading. Consequently, we can reframe problem (15) as a minimization problem for the total offloading cost:

$$\min_{\mathcal{F}} V(\mathcal{X},\mathcal{F}) \tag{18a}$$

$$\text{subject to} \quad (12b), (12c), (12d), (12e), (12f) \tag{18b}$$

Moreover, by leveraging equations (8), (9), and (17), we arrive at:

$$V(\mathcal{X},\mathcal{F}) = \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\phi_u+\psi_u p_u}{\log_2(1+\gamma_{us})} + \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\eta_u}{f_{us}}, \tag{19}$$

where, for simplicity, we define $\phi_u = \frac{\lambda_u\beta_u^{time}d_u}{t_u^{local}W}$, $\psi_u = \frac{\lambda_u\beta_u^{energy}d_u}{E_u^{local}W}$, and $\eta_u = \lambda_u\beta_u^{time}f_u^{local}$.

Given that we're not focusing on the optimization of uplink power allocation, problem (18) can be further distilled into the *Computing Resource Allocation (CRA)* problem. The subsequent sections will delve into the specifics of making computing resource allocation and offloading decisions.

A. Computing Resource Allocation (CRA)

The task of Computing Resource Allocation (CRA) revolves around optimizing the second component on the right side of equation (19), which is mathematically articulated as:

$$\min_{\mathcal{F}} \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\eta_u}{f_{us}} \tag{20a}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{U}} f_{us} \le f_s, \quad \forall s\in\mathcal{S}, \tag{20b}$$

$$f_{us} > 0, \quad \forall u\in\mathcal{U}_s, s\in\mathcal{S}. \tag{20c}$$

It's noteworthy that constraints (20b) and (20c) exhibit convexity. We designate the objective function stated in equation (20a) as $\Lambda(\mathcal{X},\mathcal{F})$. By deriving the second-order partial derivatives of $\Lambda(\mathcal{X},\mathcal{F})$ with respect to f_{us}, we attain:

$$\frac{\partial^2\Lambda(\mathcal{X},\mathcal{F})}{\partial f_{us}^2} = \frac{2\eta_u}{f_{us}^3} > 0, \quad \forall s\in\mathcal{S}, u\in\mathcal{U}_s, \tag{21a}$$

Algorithm 1 $TSAJS$: Heuristic Joint Scheduling Scheme.

```
Input: The number of users $U$, the number of servers $S$, the number of channels $N$, the total system bandwidth $W$, the user CPU frequency matrix $F_u$, the server CPU frequency matrix $F_s$, the channel gain matrix $H$, the operator's user preference matrix $\lambda_u$, the user's delay preference matrix $\beta_u^{time}$, the user's energy preference matrix $\beta_u^{energy}$, the background noise $\sigma^2$, the chip energy consumption coefficient $\kappa$;
Output: Offloading Decision Matrix: $X$, Computing Resource Allocation Matrix of the Server: $F$, Offloading Feasibility: $J$;
1: Begin
2: Initialize parameters;
3: $T \leftarrow N$; $T_{\min} \leftarrow 10^{-9}$; $\alpha_1 \leftarrow 0.97$; $\alpha_2 \leftarrow 0.90$;
4: $L \leftarrow 30$; $count \leftarrow 0$; $maxCount \leftarrow 1.75 * L$;
5: Generate the initial solution matrix $X_{old}, F_{old}, J_{old}$ that satisfies the constraint conditions;
6: $X \leftarrow X_{old}, F \leftarrow F_{old}, J \leftarrow J_{old}$;
7: while $T > T_{min}$ do
8:     $I \leftarrow 0$
9:     while $L > I$ do
10:        Use Algorithm 2 to obtain a neighborhood solution $X_{new}$ for $X_{old}$;
11:        According to $X_{new}$, use Eq. (22) to generate $F_{new}$;
12:        According to $X_{new}$, use Eq. (24) to generate $J_{new}$;
13:        $\delta = J_{new} - J_{old}$;
14:        Generate a random number $rand \in [0, 1]$;
15:        if $\delta > 0$ then
16:            $J_{old} \leftarrow J_{new}$, $X_{old} \leftarrow X_{new}$, $F_{old} \leftarrow F_{new}$;
17:            if $J > J_{\text{old}}$ then
18:                $J \leftarrow J_{\text{new}}, X \leftarrow X_{\text{new}}$, $F \leftarrow F_{\text{new}}$;
19:            end if
20:        else if $\exp(\delta/T) > rand$ then
21:            $J_{old} \leftarrow J_{new}$, $X_{old} \leftarrow X_{new}$, $F_{old} \leftarrow F_{new}$;
22:            $count \leftarrow count + 1$;
23:        end if
24:        $I \leftarrow I + 1$;
25:    end while
26:    if $count < maxCount$ then
27:        $T \leftarrow T \times \alpha_1$;
28:    else
29:        $T \leftarrow T \times \alpha_2$; $count \leftarrow 0$;
30:    end if
31: end while
32: Output $\mathcal{X}$, $\mathcal{F}$, $\mathcal{J}$;
33: End
```

$$\frac{\partial^2 \Lambda(\mathcal{X}, \mathcal{F})}{\partial f_{us} \partial f_{vw}} = 0, \quad \forall (u, s) \neq (v, w). \tag{21b}$$

Evidently, the Hessian matrix corresponding to the objective function in Eq. (20a) is diagonal, with all diagonal entries being strictly positive, indicating its positive definiteness. This confirms that Eq. (20) constitutes a convex optimization problem, which can be tackled through the application of the Karush-Kuhn-Tucker (KKT) conditions. Based on this insight, we propose the subsequent lemma.

Algorithm 2 $GetNeighborhood$: Obtain a neighborhood solution X_{new} of X_{old}.

```
Input: Use the input of Algorithm 1 and $X_{old}$;
Output: The new solution $X_{new}$;
1: Begin
2: $X_{\text{new}} \leftarrow X_{\text{old}}$;
3: Select a target user $u$ randomly;
4: Find the corresponding server $s$ and sub-channel $j$ for $u$ in $X_{\text{old}}$;
5: Generate a random number $rand$;
6: if $rand > 0.2$ then
7:     if $rand < 0.75$ then
8:         Randomly select another server $s_{other}$ different from $s$;
9:         Find a free sub-channel $j_{other}$ for $s_{other}$, or allocate one randomly if none are free;
10:        $X_{\text{new}}(u, s_{other}, j_{other}) \leftarrow 1$;
11:        $X_{\text{new}}(u, s, j) \leftarrow 0$;
12:    else if $K > 1$ then
13:        Find a free sub-channel $j_{other}$ for $s$ different from $j$, or allocate one randomly if none are free;
14:        $X_{\text{new}}(u, s, j_{other}) \leftarrow 1$;
15:        $X_{\text{new}}(u, s, j) \leftarrow 0$;
16:    end if
17: else if $rand > 0.05$ then
18:    Randomly select another user $u_{other}$;
19:    Swap the server and sub-band assignments between $u$ and $u_{other}$ in $X_{\text{new}}$;
20: else
21:    $X_{\text{new}}(u, s, j) \leftarrow 1 - X_{\text{new}}(u, s, j)$;
22: end if
23: Output $X_{new}$;
24: End
```

Lemma: The optimal allocation of computing resources f_{us}^* for the problem outlined in Eq. (20) and the resultant optimal objective function $\Lambda(\mathcal{X}, \mathcal{F}^*)$ are expressed as follows:

$$f_{us}^* = \frac{f_s \sqrt{\eta_u}}{\sum_{u \in \mathcal{U}_s} \sqrt{\eta_u}}, \quad \forall s \in \mathcal{S}, u \in \mathcal{U}_s, \tag{22}$$

$$\Lambda(\mathcal{X}, \mathcal{F}^*) = \sum_{s \in \mathcal{S}} \frac{1}{f_s} \left(\sum_{u \in \mathcal{U}_s} \sqrt{\eta_u} \right)^2. \tag{23}$$

Proof: For a detailed proof, please refer to the appendix in reference [37].

B. Unified Task Offloading Scheduling and Resource Allocation Approach

In the preceding discussions, for a specified task offloading strategy $\mathcal{X}$, we delved into the allocation of radio and computing resources. Based on Eqs. (15), (16), (19), and (23), we have derived:

$$J^*(\mathcal{X}) = \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}_s} \lambda_u(\beta_u^{time} + \beta_u^{energy}) - \Gamma(\mathcal{X}) - \Lambda(\mathcal{X}, \mathcal{F}^*) \tag{24}$$

where $\Lambda(\mathcal{X}, \mathcal{F}^*)$ is determined using the closed-form formula given in equation (23). By utilizing equation (24), we can rephrase the Task Offloading (TO) challenge from equation (14) as follows:

$$\max_{\mathcal{X}} \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}_s} \lambda_u(\beta_u^{time} + \beta_u^{energy}) - \Gamma(\mathcal{X}) - \Lambda(\mathcal{X}, \mathcal{F}^*) \tag{25a}$$

$$\text{s.t.} \quad x_{us}^j \in \{0,1\}, \quad \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}, \tag{25b}$$

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{N}} x_{us}^j \leq 1, \quad \forall u \in \mathcal{U}, \tag{25c}$$

$$\sum_{u \in \mathcal{U}} x_{us}^j \leq 1, \quad \forall s \in \mathcal{S}, j \in \mathcal{N}. \tag{25d}$$

Considering the combinatorial complexity of the Task Offloading (TO) challenge, finding an optimal solution within a polynomial timeframe is notably arduous. A conventional approach to tackle Eq. (25) involves an exhaustive search, which iterates through all potential task offloading strategies. While exhaustive search guarantees the optimal solution, the total number of candidate strategies is 2^n, where $n = S \times U \times N$, making this method impractical for large-scale problems. Another straightforward method is the greedy search, which, despite its low complexity, often converges to local optima rather than the global optimum. In paper [37], a novel meta-heuristic approach, hJTORA, was introduced to address this type of problem. hJTORA is capable of identifying a more favorable task offloading strategy with reduced complexity. However, as the number of users scales up, hJTORA cannot guarantee the optimal solution, and its execution may still be time-consuming.

To overcome the above shortcomings, we propose a joint scheduling scheme based on threshold triggered simulated annealing (TSAJS), which enables to find the local optimal task offloading decision scheme in polynomial time. We first randomly generate an initial set of solutions that satisfy the constraints and then start the simulated annealing operation. **Algorithm 2** is used at each search to obtain a set of neighborhood new solutions of the old solution. If the new solution is better than the old one, the update operation is performed; otherwise, the deteriorated new solution is accepted according to a certain probability. When the number of times the new solution is inferior to the old solution accumulates up to a certain threshold, we accelerate the rate of temperature decrease. In summary, our proposed heuristic for task offloading scheduling is given in **Algorithm 1**.

V. Experimental Evaluation

Simulation results are given to evaluate the performance of the proposed joint scheduling scheme based on threshold triggered simulated annealing,referred to as TSAJS. Similar to the paper [37], we examine a multi-cellular network comprising several hexagonal cells, each centered around a base station. The distance between adjacent base stations is maintained at 1 kilometer. It is presupposed that both the user and the base station employ a single antenna for the uplink transmission and reception processes, respectively. The uplink channel gain is derived from a path loss model that is contingent upon the distance, specifically $L[\text{dB}] = 140.7 + 36.7 \log_{10} d[\text{km}]$, with the lognormal shadowing standard deviation fixed at 8 dB. Unless specified otherwise, the simulations predominantly consider a scenario with S = 9 cells, with the user's transmission power configured to P_u = 10 dBm. Moreover, the system bandwidth is configured to B = 20 MHz, and the variance of the background noise is hypothesized to be $\sigma^2 = -100$ dBm.

Considering computational resources, we presume that the processing power per Multi-Access Edge Computing (MEC) server is f_s = 20 GHz and per user is f_u = 1 GHz . The energy consumption factor is set to $\kappa = 5 \times 10^{-27}$. For computational tasks, we select a standard input size of d_u= 420 KB and set preference parameters β_u^{time} = 0.5 , β_u^{energy} = 0.5 , and λ_u= 1 , $\forall u \in U$. Unless specified, the number of subbands is typically set to 3. Users are randomly and uniformly distributed across the network's coverage area.

The system utility performance of our proposed TSAJS scheme in this paper is compared with the following baselines:

- Exhaustive Method: Utilizing a brute-force strategy, this method explores every possible decision among 2^n options to determine the most effective offloading scheduling solution. Given its substantial computational demands, our performance assessment is limited to a confined network setting.
- hJTORA: A heuristic algorithm designed to address the TO problem, which obtains a suboptimal solution within polynomial time [37].
- Greedy Offloading Method: All permissible tasks, up to the limit set by the base stations, are offloaded. Users are assigned to sub-bands in a prioritized manner, favoring those with the strongest signal strength.
- LocalSearch: Continuously search for neighboring states of the current state when users offload tasks, and accept better neighboring states to gradually improve the quality of the solution. The search stops when the algorithm converges or reaches the maximum number of iterations.

A. *Suboptimality Analysis of the TSAJS Algorithm*

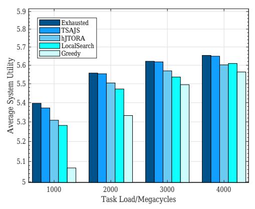


Fig. 3. Comparison of average system utility among five schemes.

To validate the near-optimal nature of the TSAJS solution obtained by Algorithm 1, we first compared it with the global optimal solution derived from an exhaustive search method, and further conducted a comparative analysis with three baseline strategies, with the specific results shown in Figure 3. Given that an exhaustive search examines all possible offloading scheduling schemes, the computation time becomes extremely lengthy when dealing with large-scale variables. Therefore, we conducted experiments in a smaller network environment consisting of U=6 users evenly distributed within the coverage of S=4 cells, each equipped with N=2 sub-bands. With user task loads w_u set at 1000, 2000, 3000, and 4000 Megacycles respectively, we calculated the corresponding average system utility for each scheme and provided the 95% confidence interval (CI). The experimental results demonstrate that, across all test scenarios, the average system utility achieved by the TSAJS algorithm is almost comparable to that of the exhaustive search method, while also achieving approximately 0.9%, 1.49%, and 4.14% average performance improvements compared to the hJTORA, LocalSearch and Greedy strategies, respectively. This fully proves that the performance of the TSAJS algorithm proposed in this paper is very close to the optimal solution obtained by the exhaustive search method and significantly superior to other baseline strategies. Additionally, we observed that the system utility of all schemes gradually increases with the increase in task load.

B. Analysis of the Impact of Changes in User Numbers

Figure 4 displays the system utility performance as the number of users unloading tasks varies, with a particular comparison between two settings of L=10 and L=30. Given the characteristics of task offloading scenarios, an increase in the number of users results in a reduction of bandwidth allocated to each user. According to the figure, the TSAJS strategy consistently exhibits the best performance, and as the computational demand of tasks increases, the system utility of all strategies shows a significant improvement. This trend can be explained by the fact that, as tasks become more computationally intensive, users find greater advantage in delegating these tasks to the MEC server. Upon closer examination, it becomes evident that the system's efficiency initially shows an upward trend with an increasing user base. Nonetheless, when the user count surpasses a particular threshold, the system's efficiency starts to deteriorate. This decline is attributable to the heightened competition among a large user base for wireless bandwidth and computational resources necessary for task offloading. Consequently, this results in escalated costs associated with task transmission and processing on the MEC server, thereby undermining the advantages of offloading. Notably, in the scenario where L=30, the TSAJS strategy still achieves continuous growth in system utility, thanks to its ability to explore better solutions and ensure more reasonable and efficient resource allocation.

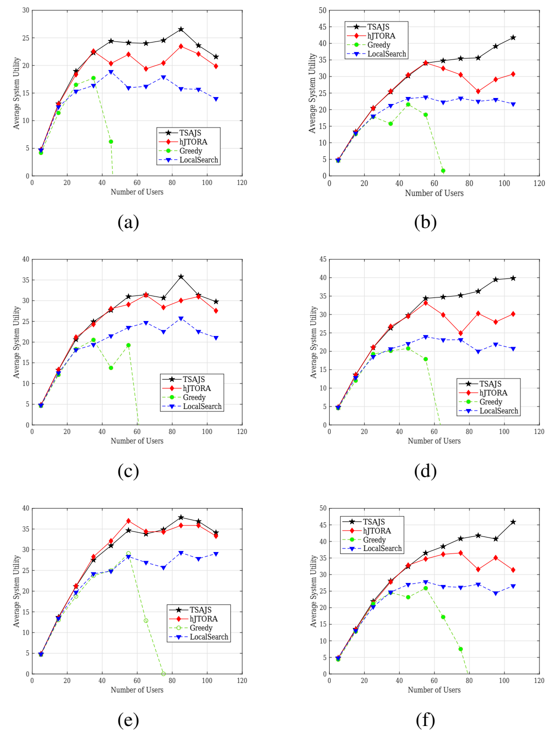


Fig. 4. A comparison of the average system utility for four different schemes across three distinct task workloads and varying user scales. (a) w_u=1000 Megacycles, L=10. (b) w_u=1000 Megacycles, L=30. (c) w_u=2000 Megacycles, L=10. (d) w_u=2000 Megacycles, L=30. (e) w_u=3000 Megacycles, L=10. (f) w_u=3000 Megacycles, L=30.

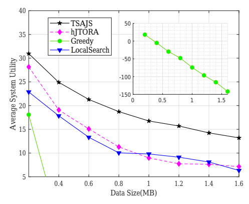


Fig. 5. The average system utility varies with the size of the task data.

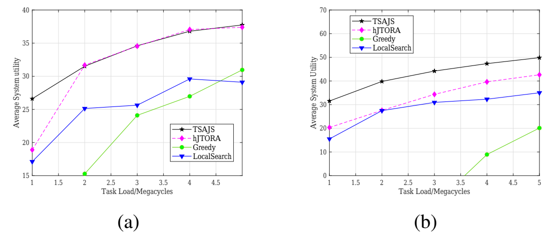
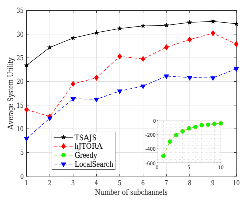


Fig. 6. Comparison of the average system utility of four schemes with varying task workloads while the number of users remains fixed. (a) U=50. (b) U=90.

C. Analysis of the Impact of Task Configuration Changes

Figure 5 reveals the relationship between average system utility and task data size. It is observed that as the task input size gradually increases, the average system utility of various


(a)

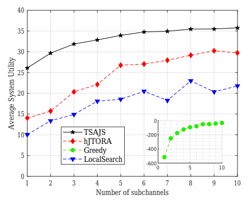
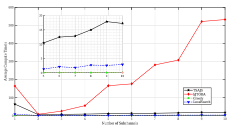

(b)

Fig. 7. Comparison of the average system utility of four schemes with varying numbers of sub-channels. (a) L=30. (b) L=50.


(a)

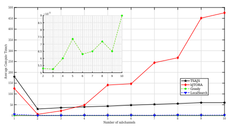
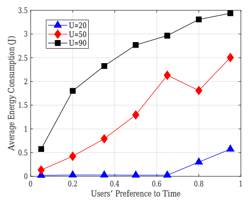

(b)

Fig. 8. Average computation time as a function of the number of sub-channels. (a) L=10. (b) L=50.

schemes exhibits a decreasing trend. Figure 6 provides an in-depth comparison of the average system utility performance of four schemes under different task workloads, with the number of users remaining constant. Through the contrast between two scenarios, (a) U=50 and (b) U=90, we can clearly see that the average system utility of all schemes increases continuously with the increase in task workload. Based on the combined analysis of Figure 5 and Figure 6, we can draw an important conclusion: Tasks with smaller input sizes but higher workloads benefit more from being offloaded to MEC servers, compared to tasks with larger input sizes but lower workloads. Furthermore, the performance of the TSAJS scheme proposed in this paper also demonstrates a similar trend: its performance gain gradually increases with the increase in task workload, while it decreases with the increase in task input size.

D. Analysis of the Impact of Changes in the Number of Sub-Channels

Based on the analysis of Figures 7 and 8, we can draw the following conclusions: As the number of sub-channels increases, the average system utility demonstrates a trend of first increasing and then decreasing. This is primarily because excessive sub-channels may lead to channel idleness, thereby reducing the offloading utility. Among the schemes, the TSAJS scheme exhibits the best performance when the number of sub-channels is 30 and 50, showcasing its superior capabilities. On the other hand, with the increase in the number of sub-channels, the average computation time also extends, attributed to the expansion of the search scope. Notably, the computation time of the hJTORA scheme increases more significantly, while the average computation time of the LocalSearch and Greedy schemes remains relatively stable. This stability is mainly attributed to their adoption of a fixed search approach.


(a)

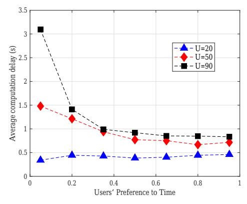

(b)

Fig. 9. Impact of user preferences. (a) Average Energy Consumption. (b) Average computation delay.

E. Analysis of the Impact of User Preferences

In Figures 9(a) and (b), we conducted experiments to evaluate the performance of TSAJS under three different user scales. In the experimental design, we systematically adjusted the user's time preference parameter, β_u^{time}, which ranged from 0.05 to 0.95 in increments. Correspondingly, the user's energy preference, β_u^{energy}, was dynamically adjusted according to the relationship $\beta_u^{energy} = 1 - \beta_u^{time}$, ensuring that their sum remained constant at 1. Through this setup, we comprehensively observed the average performance of all users in terms of time consumption and energy consumption. The experimental results demonstrated that as the value of β_u^{time} gradually increased, users tended to prioritize time efficiency, leading to a significant reduction in average time consumption. However, this temporal optimization was not without a trade-off; it came at the expense of increased energy consumption. In other words, as users pursued faster task completion, they had to accept higher levels of energy consumption.

VI. Conclusions

In this paper, we present TSAJS, an efficient multi-server joint task scheduling scheme for MEC. We model the task offloading and resource allocation problem as a MINLP problem with the goal of maximizing user offloading gain. TSAJS decomposes this complex problem into two stages: offloading decision and computational resource allocation. For offloading decision, a heuristic algorithm combining threshold-triggering and simulated annealing is used to avoid local optima. Optimal resource allocation is then derived using KKT conditions. TSAJS is both efficient and flexible, finding near-optimal solutions within polynomial time. Experimental results demonstrate that it performs close to the optimal solution and significantly outperforms traditional methods. As a promising heuristic approach, TSAJS enhances MEC offloading effectiveness and addresses the increasing demands of mobile users.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 62422208, 62202276, 62232010), in part by Shandong Science Fund for Excellent Young Scholars (No. 2022HWYQ-038), in part by Shandong Science Fund (No. 2023TSGC0105, 2024TSGC0228), in part by Ministry of Industry and Information Technology of China: Cloud-Data Integrated Machine for Cloud-Edge Collaboration.

REFERENCES

[1] Z. Hu, J. Niu, T. Ren, and M. Guizani, "Achieving fast environment adaptation of drl-based computation offloading in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 6347–6362, 2024.

[2] S. Chu, C. Gao, M. Xu, K. Ye, Z. Xiao, and C. Xu, "Efficient multi-task computation offloading game for mobile edge computing," *IEEE Transactions on Services Computing*, vol. 17, no. 1, pp. 30–46, 2024.

[3] W. Jiang, D. Feng, Y. Sun, G. Feng, Z. Wang, and X.-G. Xia, "Joint computation offloading and resource allocation for d2d-assisted mobile edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 1949–1963, 2023.

[4] J. Mei, Z. Tong, K. Li, L. Zhang, and K. Li, "Energy-efficient heuristic computation offloading with delay constraints in mobile edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 4404–4417, 2023.

[5] P. Teymoori and A. Boukerche, "Latency-constrained dynamic computation offloading in mobile edge computing using multi-agent reinforcement learning," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 3688–3693.

[6] Z. Zhang, H. Zhouand, L. Zhao, and V. C. M. Leung, "Digital twin assisted computation offloading and service caching in mobile edge computing," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 1296–1297.

[7] X. Pang, Z. Wang, J. Li, R. Zhou, J. Ren, and Z. Li, "Towards online privacy-preserving computation offloading in mobile edge computing," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1179–1188.

[8] R. Zhang and C. Zhou, "A computation task offloading scheme based on mobile-cloud and edge computing for wbans," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 4504–4509.

[9] Z. Hu, J. Niu, T. Ren, B. Dai, Q. Li, M. Xu, and S. K. Das, "An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 669–683, 2022.

[10] G. Interdonato and S. Buzzi, "Joint optimization of uplink power and computational resources in mobile edge computing-enabled cell-free massive mimo," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1804–1820, 2024.

[11] J. Du, H. Wu, M. Xu, and R. Buyya, "Computation energy efficiency maximization for noma-based and wireless-powered mobile edge computing with backscatter communication," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 6954–6970, 2024.

[12] W. Ma and L. Mashayekhy, "Video offloading in mobile edge computing: Dealing with uncertainty," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 251–10 264, 2024.

[13] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[14] L. Liu, B. Sun, Y. Wu, and D. H. K. Tsang, "Latency optimization for computation offloading with hybrid noma–oma transmission," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6677–6691, 2021.

[15] S. Li, W. Sun, Y. Sun, and Y. Huo, "Energy-efficient task offloading using dynamic voltage scaling in mobile edge computing," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 588–598, 2021.

[16] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5g ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[17] T. X. Tran and D. Pompili, "Dynamic radio cooperation for user-centric cloud-ran with computing resource sharing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2379–2393, 2017.

[18] I. Nokia, "Increasing mobile operators value proposition with edge computing," *Technical Brief*, 2013.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[20] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[21] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[22] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[23] D. Cao, M. Wu, N. Gu, R. S. Sherratt, U. Ghosh, and P. K. Sharma, "Joint optimization of computation offloading and resource allocation considering task prioritization in isac-assisted vehicular network," *IEEE Internet of Things Journal*, 2024.

[24] W. Ma and L. Mashayekhy, "Video offloading in mobile edge computing: Dealing with uncertainty," *IEEE Transactions on Mobile Computing*, 2024.

[25] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4000–4015, 2022.

[26] R. van der Meulen *et al.*, "What edge computing means for infrastructure and operations leaders," *Smarter with Gartner*, 2018.

[27] OpenFog - OPC Foundation, "Markets & collaboration," https://opcfoundation.org/markets-collaboration/openfog/, 2021, accessed: May 5, 2021.

[28] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186 080–186 101, 2020.

[29] S. Zhang, N. Yi, and Y. Ma, "A survey of computation offloading with task types," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[30] F. Zhang, G. Han, L. Liu, M. Martínez-García, and Y. Peng, "Deep reinforcement learning based cooperative partial task offloading and resource allocation for iiot applications," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2991–3006, 2022.

[31] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2018.

[32] L. Liu, B. Sun, Y. Wu, and D. H. Tsang, "Latency optimization for computation offloading with hybrid noma–oma transmission," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6677–6691, 2021.

[33] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[34] J. Gao, R. Chang, Z. Yang, Q. Huang, Y. Zhao, and Y. Wu, "A task offloading algorithm for cloud-edge collaborative system based on lyapunov optimization," *Cluster Computing*, vol. 26, no. 1, pp. 337–348, 2023.

[35] A. Heidari, M. A. J. Jamali, N. J. Navimipour, and S. Akbarpour, "A qos-aware technique for computation offloading in iot-edge platforms using a convolutional neural network and markov decision process," *IT Professional*, vol. 25, no. 1, pp. 24–39, 2023.

[36] J. Huang, J. Wan, B. Lv, Q. Ye, and Y. Chen, "Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2500–2511, 2023.

[37] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[38] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USA: USENIX Association, 2010, p. 4.

[39] E. Dahlman, S. Parkvall, and J. Sköld, "Chapter 3 - ofdm transmission," in *4G LTE/LTE-Advanced for Mobile Broadband*, E. Dahlman, S. Parkvall, and J. Sköld, Eds. Oxford: Academic Press, 2011, pp. 27–44. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123854896000035

[40] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2706–2716, 2013.

[41] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach." 1987.