Two-stage Evolutionary Search for Efficient Task Offloading in Edge Computing Power Networks

Qunjian Chen, Chen Yang, *Member, IEEE*, Shulin Lan, *Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, and Yan Zhang, *Fellow, IEEE*

***Abstract*—In this paper, we introduce the concept of edge computing power network (EdgeCPN) as a new paradigm to facilitate elastic integration and flexible scheduling of computing resources for task offloading in CPNs. Previous studies mainly focused on scheduling computing resources in the vertical dimension and may not effectively consider the computing resources selection in CPNs with increasingly diverse computing resources, which results in inefficient and unstable computing resource scheduling performance for task offloading. In this paper, we design an on-demand computing resource scheduling model to enable efficient task offloading in EdgeCPNs. To improve the search efficiency and stability, we decouple the search for task offloading problems in EdgeCPNs into two stages and present an two-stage evolutionary search scheme (TESA). In stage-1, TESA first optimizes computing resources selection by searching a computing resources subset depending on the user budget, with the objective of maximizing the total gain. In stage-2, TESA jointly optimizes task offloading decisions and computing resources allocations based on the subset found in stage-1, with the objective of minimizing total delay. Numerical results confirm that the proposed scheme significantly enhances the efficiency and stability of the computing resources scheduling performance for task offloading in EdgeCPNs.**

***Index Terms*—Computing power network, mobile edge computing, task offloading, evolutionary optimization, computing resources scheduling.**

I. Introduction

For compute-intensive and delay-sensitive smart applications, task offloading is an effective strategy to enhance application performance and improve user experience [1]–[4]. Popular cloud computing and mobile edge computing (MEC) paradigms have their own advantages and limitations [5], [6]. Cloud computing provides users with sufficient computing capability on remote servers, but suffers from the drawbacks of high latency and low reliability [7]–[10]. MEC serves as a complementary solution that brings computing capability from the nearby edge server [11], [12], which contributes to reducing latency overhead and providing privacy protection for various applications [13]–[16]. Unfortunately, the edge server can easily become overloaded and application performance can be severely degraded. Most studies have considered either the device-edge two-level task offloading architectures [17]–[19] or the device-edge-cloud three-level task offloading architectures [20]–[23], as shown in Fig. 1. However, these studies mainly focus on computing resource scheduling in the vertical dimension and may not effectively consider the horizontal collaboration among computing resources.

The convergence of networking and computing is a promising trend in network evolution [24], [25]. As an innovative network model, the computing power network (CPN) has been proposed to facilitate adequate exploitation of heterogeneous and geographically distributed computing resources. Through the integration and scheduling of fragmented computing resources across different MEC sites, computing resources sharing can be effectively realized on a broader scale in CPNs. **With the growing diversity of computing resources owned by various organizations with different features, the problem of scheduling computing resources for task offloading in CPNs becomes increasingly complex.** Making the optimal use of fragmented computing resources for efficient task offloading in CPNs has become a significant challenge.

Previous studies have generally addressed the task offloading problem in CPNs by searching the task offloading decisions and computing resources allocations [26], [27], but have rarely considered the critical aspect of computing resources selection. **First, they usually assume that all computing resources in CPNs are available for task offloading.** However, this does not align with the actual situation, as **access to computing resources is usually constrained by limited user budget**. It also inevitably leads to a decrease in search efficiency, because the search space for task offloading problems significantly increases with the expansion of available computing resources in CPNs. Therefore, integrated search schemes for identifying task offloading decisions and computing resources allocations can frequently become entangled with sub-optimal solutions, resulting in inefficient computing resources scheduling performance for task offloading in CPNs. **Second, they may not effectively compare the different computing resources in CPNs, with difficulty in wisely selecting the most appropriate computing resources to satisfy the performance and cost requirements of users [23].** For

This work was supported in part by the National Key R&D Program of China under Grant 2021YFB1715700; in part by the National Natural Science Foundation of China under Grant 62103046, 72201266, 72192843, and 72192844; in part by the Fundamental Research Funds for the Central Universities under Grant E1E40805X2 and 2023CX01020. *(Corresponding author: Chen Yang, Shulin Lan).*

Qunjian Chen is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (Email: ChenQJ@outlook.com).

Chen Yang and Liehuang Zhu is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (Email: yangchen666@bit.edu.cn; liehuangz@bit.edu.cn).

Shulin Lan is with the School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100081, China (Email: lanshulin@ucas.ac.cn).

Yan Zhang is with the Department of Informatics, University of Oslo, 0316 Oslo, Norway, and also with the Simula Metropolitan Center for Digital Engineering, 0167 Oslo, Norway (e-mail: yanzhang@ieee.org).

example, given that computing resources in CPNs may vary in computing capacities, network connections, and service costs, the effectiveness of task offloading can be adversely affected by instability in computing resources scheduling performance if the combination of computing resources is randomly selected without considering the benefits they can provide. **Third, most studies may struggle to provide an expandable task offloading architecture to meet varying computing power requirements.** Throughout the task offloading process, the computing resources available for the architecture will keep invariable, and the corresponding scheduling algorithm developed for the architecture follow a fixed task offloading model, resulting in weak generality.

To address the above challenges, **we introduce the concept of edge computing power network (EdgeCPN), which enables a fairly elastic integration and flexible scheduling of fragmented computing resources in the CPN in response to local user demand.** An EdgeCPN is designed as an autonomous unit of computing power (within the CPN), where computation tasks offloaded from mobile devices can be processed through site-autonomy or site-collaboration modes. It operates independently in site-autonomy mode, or collaborates with other EdgeCPNs in site-collaboration mode. The site-collaboration mode allows an MEC site to collectively utilize fragmented computing resources from other MEC sites to meet the computing resources requirements of users. The main contributions of this paper can be summarized as follows:

- We propose a new paradigm, the EdgeCPN, and establish a flexible task offloading architecture to jointly schedule computing resources across different MEC sites. This architecture allows the site-autonomy mode to be extended to the site-collaboration mode, efficiently alleviating the problem of insufficient computing resources.
- To identify the most suitable computing resources for computation tasks, we develop an on-demand computing resources scheduling model for task offloading in EdgeCPNs. This model facilitates the optimal selection of a subset of computing resources from the network based on user requirement.
- For efficient and stable resource scheduling performance, we decouple the search for task offloading problems in EdgeCPNs into two stages and propose a two-stage efficient evolutionary search scheme (TESA). By optimizing the computing resources selection in stage-1 to improve the corresponding searches in stage-2, the total delay of task offloading can be significantly reduced.

The rest of this paper is organized as follows. Section II discusses the related work. Section III presents the system model and problem formulation. Section IV introduces the proposed scheme in detail. Section V gives the analysis of numerical results. Section VI concludes this paper.

II. Related Work

A. *Two or Three-level Task Offloading Architecture*

Some efforts have been made to enhance computing resources scheduling in MEC networks [28], [29], aimed at dealing with task offloading problem and improving application performance. Wang et al. [30] investigated a joint computation task, spectrum, and transmission power allocation problem in MEC networks, the objective of which is to adjust the strategies of computation tasks and computing resources allocation according to changes in computation tasks to minimize computation and transmission delays among all users. Zhang et al. [31] simultaneously considered the task offloading problems in both single-cell and multi-cell MEC networks, taking into account the residual energy of smart devices' batteries and jointly optimizing the allocation of communication and computing resources under sensitive delay conditions. However, these studies primarily resolve around the device-edge two-level task offloading architectures within a single MEC site. If the computing resources limitation of the edge server is not taken into account, it may lead to

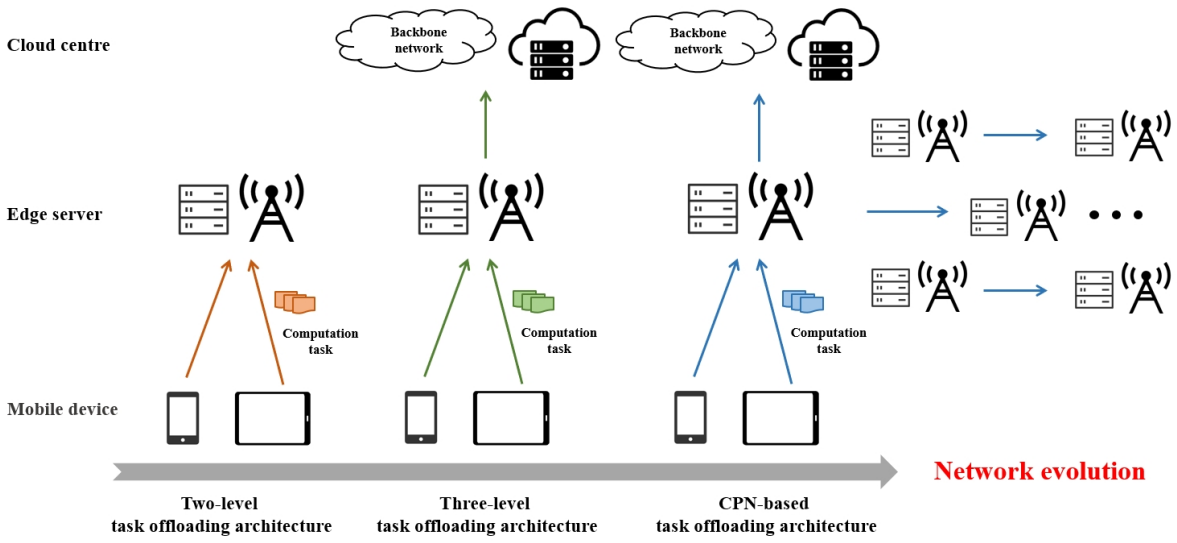

Fig. 1. Examples of task offloading architectures.

fierce computing resources competition among computation tasks when the edge server is overloaded.

To alleviate this problem, many studies have considered the cloud-assisted device-edge-cloud three-level task offloading architectures, which can reduce the load on the edge server by offloading part of the computation tasks to the cloud center. Motivated by the synergistic effects of cloud center, static fog and mobile fog in handling time-critical tasks, Liu et al. [32] proposed a two-layer vehicular fog computing architecture that collaboratively offloads computation tasks to different nodes based on their classification characteristics. Feng et al. [33] assumed that computation tasks can be partitioned into multiple subtasks and can be executed on local devices, MEC servers and cloud servers. They formulated the problem of task partitioning and user association in MEC systems and proposed a dual decomposition-based approach and a matching-based approach to solve it. However, cloud-related approaches introduce the risk of high communication latency. As computation tasks are executed across the cloud centre and the edge server, they need to traverse core network and backbone network, leading to significant differences in application response time. While both task offloading architectures consider the vertical computing resources scheduling, they may not make efficient utilization of computing resources in the horizontal dimension.

B. *MEC or CPN-based Task Offloading*

There have been several studies that explored the use of neighboring MEC sites' computing resources to collaboratively improve task offloading, offering an effective solution to mitigate the above challenges. Peng et al. [34] considered a multi-server multi-user multi-task computation offloading scenario in collaborative edge and cloud computing networks. They formulated a constrained multi-objective computation offloading model considering time and energy consumption and proposed a multi-objective optimization algorithm based on a push-pull search framework to search solutions. Laili et al. [35] studied how to collaboratively execute interconnected manufacturing and computation tasks in manufacturing units, cloud resources, and edge resources. They proposed a discretized soft actor-critic configured differential evolution algorithm to obtain a robust scheme to the cloud-edge-device collaborative task scheduling problem.

By integrating networking and computing, this concept can be extended into a universal service: a unified supply of computing resources based on CPNs. Tang et al. [24] established a computing resources sharing platform in CPNs that connects users and computing resources providers for facilitating efficient computing resources scheduling, which reduces the edge server's reliance on the cloud center. Yao et al. [36] proposed a computing-aware routing protocol in CPNs to facilitate the coordinated optimization of network and computing resources, where service requests can be scheduled to the optimal service node along the network path. Hao et al. [37] studied the task offloading problems in CPNs under time-continuous conditions to further optimize the overall waiting delay by reducing decision waiting time. Lu et al. [26] modeled the wireless CPN for computing resources scheduling in multiple MEC systems. They solved the task transfer problem in the wireless CPN by unifying the scheduling of computing resources from mobile devices and MEC sites. In addition, Sun et al. [38] designed a task and resource-aware federated learning model in WCPNs, which jointly optimizes the computing strategies and collaboration mechanisms for computing nodes to reduce the energy consumption. However, most of the existing studies have not effectively considered the problem of computing resources selection for task offloading in CPNs that may lead to inefficient and unstable computing resources scheduling performance.

In summary, compared to existing work, our approach practically addresses the resource selection process from the vast and diverse resource pool of the CPN under the constraints of a limited user budget for task offloading. The introduction of the EdgeCPN concept supports both the site-autonomy and site-collaboration modes, enabling the utilization of fragmented computing resources outside the current EdgeCPN in CPNs. This provides a flexible mechanism for on-demand scheduling of computing resources. Then for matching the most appropriate computing resources for computation tasks and reducing the total delay of task offloading, we divide the search for task offloading problems in EdgeCPNs into two stages and propose a two-stage efficient evolutionary search scheme, considering both the performance and cost requirements of users. Our approach has the advantage of elastic integration and flexible scheduling of fragmented computing resources throughout the CPN in response to local user demand.

III. System Model and Problem Formulation

A. *On-demand Computing Resources Scheduling*

As shown in Fig. 2, we design an on-demand computing resources scheduling model for task offloading in EdgeCPNs,

TABLE I
MAIN NOTATIONS.

$\mathcal{M}$	The mobile device set in the current MEC site
$\mathcal{N}_m$	The computation task set of mobile device m
$d_{m,n}$	The input data size of computation task n generated from mobile device m
$f_{m,n}$	The required CPU cycles of computation task n generated from mobile device m
t_m	The maximum delay tolerance for mobile device m
c_m^{md}	The computing power of mobile device m
R_m	The transmission rate from mobile device m to the edge server
p_m	The transmission power from mobile device m to the edge server
h_m	The channel gain between mobile device m and the edge server
σ^2	The noise power
ω	The pathloss constant
λ	The pathloss exponent
c^{es}	The computing power of the edge server
$\mathcal{K}$	The set of optional MEC sites in EdgeCPNs
c_k^{ms}	The available computing power of external MEC site k
r_k	The transmission rate from the current MEC site to external MEC site k
u_k	The rental cost of external MEC site k
U	The user budget
$\mathcal{S}$	A subset of computing resources selected from $\mathcal{K}$
$Q(\mathcal{S})$	The total gain of computing resources subset $\mathcal{S}$
$C(\mathcal{S})$	The total cost of computing resources subset $\mathcal{S}$
α, β	Two weight factors.

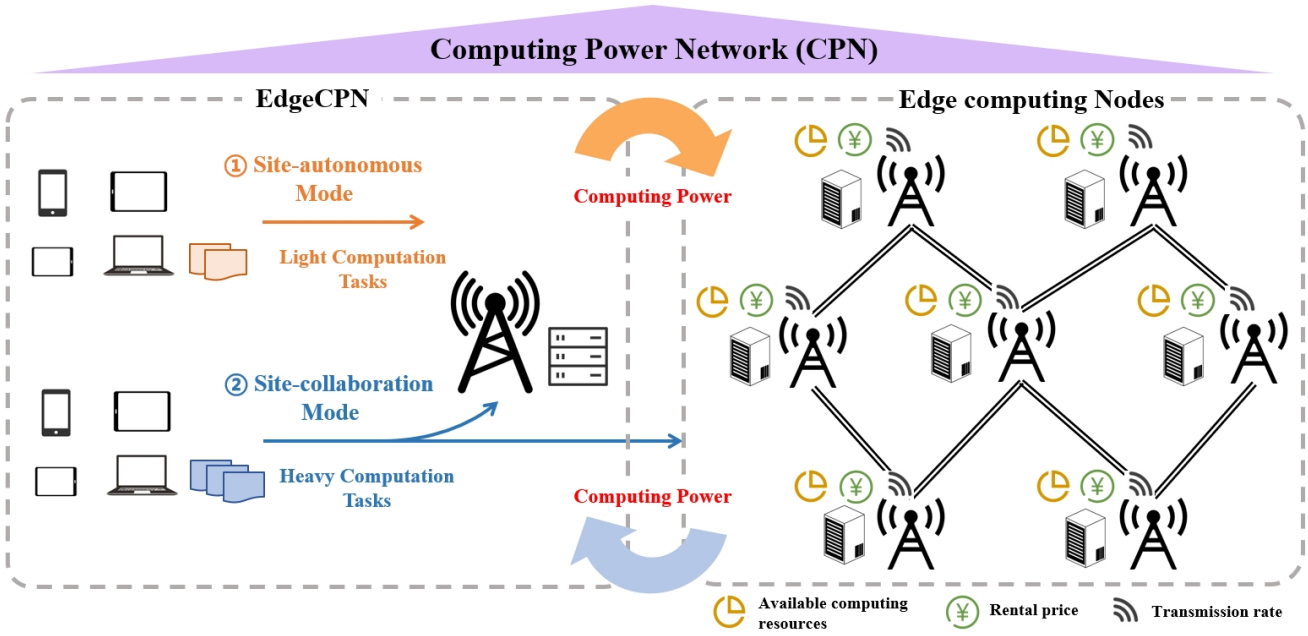

Fig. 2. Task offloading in EdgeCPNs.

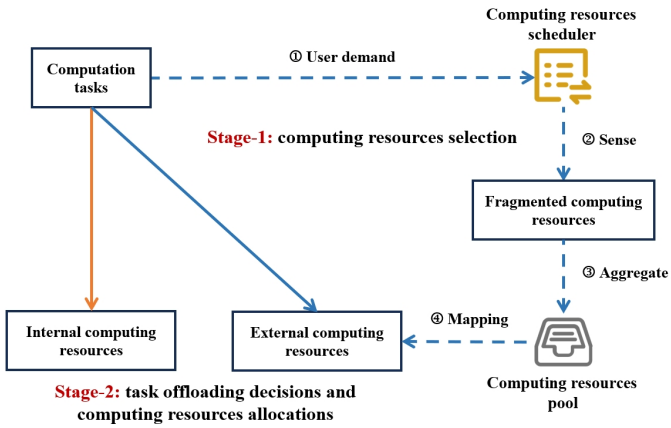

Fig. 3. Computing resources scheduling process.

which aims to search for the most appropriate computing resources from the network to match user demand.

Mobile devices within the current MEC site's coverage can offload their computation tasks to the edge server over the wireless channel, for the purpose of improving application response time. However, with the growth of mobile devices and computation tasks, the load on one edge server may be rising rapidly, ultimately leading to unacceptable latency levels. In CPNs, there are a large number of geographically distributed MEC sites, some of which may remain underloaded or idle. Therefore, the restriction of computing resources within the current MEC sites can be relieved by integrating and scheduling the fragmented computing resources from external MEC sites. In this case, a portion of the computation tasks can be transferred to external MEC sites for execution.

To this end, an EdgeCPN is considered as an autonomous unit of computing power within the CPN, where computation tasks offloaded from mobile devices can be processed through site-autonomy or site-collaboration modes. Specifically, the site-autonomy mode refers to scheduling computing resources within the current MEC site of the EdgeCPN to perform task offloading without relying on external MEC sites. When internal computing resources are not sufficient for computation tasks from mobile devices, it can request additional computing resources from external MEC sites according to the user demand. Therefore, task offloading in the site-collaboration mode involves the joint utilization of internal and external computing resources. Due to the diversity of external computing resources, they must be appropriately selected to improve the computing resources scheduling performance for task offloading.

As illustrated in Fig. 3, the computing resources scheduling process in EdgeCPNs is given. First, the computing resources scheduler receives user demand and builds an empty computing resources pool. Second, the computing resources scheduler senses the status of external MEC sites distributed in the network, such as their available computing resources and rental cost, as well as the transmission rates between the current MEC site and them. Third, the computing resources scheduler selects a portion of external MEC sites from the network based on the user budget. The computing resources of selected MEC sites are aggregated into the computing resources pool. Finally, the available computing resources in the computing resources pool are mapped as external computing resources, which are utilized in combination with internal computing resources to enable task offloading. The following is the detailed description of the system model and problem formulation. For easy reading, Table I summarises the main notations used in this section.

B. Site-autonomy Mode

In site-autonomy mode, the task offloading problem includes the search of the task offloading decisions and the computing resources allocations under the computing resources restriction of the current MEC site.

We use $\mathcal{M} = \{1, 2, ..., M\}$ to denote the mobile device set in the current MEC site. The mobile device m has a computation task set $\mathcal{N}_m = \{1, 2, ..., N_m\}$ that need to be executed. The input data size and the required CPU cycles of computation task n generated from mobile device m are represented as $d_{m,n}$ and $f_{m,n}$, respectively. The maximum delay tolerance for mobile device m is denoted as t_m. When the computing power of mobile device m cannot meet the latency requirements, it can initiate a request to the edge server deployed at the current MEC site to offload compute-intensive computation tasks.

Given task offloading decision $x_{m,n} \in \{0, 1\}$, where $x_{m,n} = 0$ indicates that the computation task n generated from mobile device m is executed at the mobile device m and $x_{m,n} = 1$ indicates that the computation task n generated from mobile device m is executed on the edge server. We assume computation tasks left at the mobile device m need to be executed serially, while computation tasks offloaded to the edge server can be executed in parallel. Due to the size of the computation result is much smaller than the size of the input data, the return delay of computation result is ignored [39]. Thus, the total delay of task offloading in site-autonomy mode consists of mobile device execution delay, edge server transmission delay, and edge server execution delay.

Let c_m^{md} denote the computing power of mobile device m, measured in CPU frequency. The mobile device execution delay can be calculated by:

$$T_m^{md} = \sum_{n \in \mathcal{L}^1} \frac{f_{m,n}}{c_m^{md}}; \mathcal{L}^1 = \{n | x_{m,n} = 0\} \tag{1}$$

Through wireless transmission [25], mobile device m can offload its computation tasks to the edge server. Following from Shannon's channel capacity formulation, the transmission rate from mobile device m to the edge server can be mathematically as:

$$R_m = B \log_2(1 + \frac{p_m h_m}{\sigma^2}) \tag{2}$$

where B is the bandwidth allocated to mobile device m, p_m is the transmission power, h_m is the channel gain, and σ^2 is the noise power. We assume the wireless channel to be dominated by line-of-sight component, where the channel gain can be expressed as:

$$h_m = \omega(\frac{l_0}{l_m})^{\lambda} \tag{3}$$

where ω is a pathloss constant, λ is the pathloss exponent, l_0 is a reference distance, and l_m is the distance between mobile device m and the edge server.

The edge server transmission delay can be calculated by:

$$T_m^{me} = \sum_{n \in \mathcal{L}^2} \frac{d_{m,n}}{R_m}; \mathcal{L}^2 = \{n | x_{m,n} > 0\} \tag{4}$$

Note that computation tasks need to be fully transmitted before they can be executed by the edge server.

Let c^{es} denote the computing power of the edge server that can be shared by offloaded computation tasks. Given computing resources allocation $y_{m,n} \in [0, c^{es}]$, the edge server execution delay can be expressed by:

$$T_m^{es} = \max_{n \in \mathcal{L}^2} \frac{f_{m,n}}{y_{m,n}}; \mathcal{L}^2 = \{n | x_{m,n} > 0\} \tag{5}$$

Therefore, the total delay required to complete the computation tasks of mobile device m can be calculated by:

$$T_m^{SA} = \max\{T_m^{md}; T_m^{me} + T_m^{es}\} \tag{6}$$

The task offloading problem in the site-autonomy mode involves the optimization of the following decision variables: i) task offloading decision $x_{m,n} \in \{0, 1\}$, is a binary variable; ii) computing resources allocation $y_{m,n} \in [0, c^{es}]$, is a continuous variable. With the goal of minimizing the total delay for all mobile devices, the task offloading problem in the site-autonomy mode can be defined as:

$$\mathbf{P1}: \min_{x_{m,n}, y_{m,n}} \sum_{m \in \mathcal{M}} T_m^{SA} \tag{7}$$

$$s.t. \quad x_{m,n} \in \{0, 1\}, \forall m, \forall n \tag{8}$$

$$y_{m,n} \in [0, c^{es}], \forall m, \forall n \tag{9}$$

$$T_m^{SA} \leq t_m, \forall m \tag{10}$$

$$\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} x_{m,n} y_{m,n} \leq c^{es} \tag{11}$$

C. Site-collaboration Mode

In site-collaboration mode, the search of the task offloading problem is decoupled into two stages, where stage-1 optimizes the computing resources selection, and stage-2 jointly optimizes the task offloading decisions and the computing resources allocations based on the computing resources identified in stage-1.

Stage-1: Computing Resources Selection. We use $\mathcal{K} = \{1, 2, ..., K\}$ to denote the optional computing resources set consisting of external MEC sites in EdgeCPNs. The available computing resources and the rental cost of external MEC site k are denoted as c_k^{ms} and u_k, respectively. The transmission rate from the current MEC site to external MEC site k is denoted as r_k. Considering the diversity of optional computing resources set in EdgeCPNs, the value of K is usually large, leading to a large search space for the task offloading problem and struggling to obtain stable computing resources scheduling performance for task offloading. Therefore, we first formulate the computing resources selection of the task offloading problem as a subset selection problem, and then the selected computing resources subset is used as the search space for task offloading decisions and computing resources allocations of the task offloading problem.

Let U denote the user budget, the computing resources selection of the task offloading problem aims to search a computing resources subset $\mathcal{S} = \{1, 2, ..., S\}$ from $\mathcal{K}$ with

the highest gain. The total gain of the selected computing resources subset $\mathcal{S}$ can be expressed as:

$$Q(\mathcal{S}) = \sum_{s \in \mathcal{S} \subseteq \mathcal{K}} \alpha r_s + \beta c_s^{ms} \tag{12}$$

where c_s^{ms} denotes the available computing resources of selected MEC site s, r_s denotes the transmission rate from the current MEC site to selected MEC site s, α and β are two weight factors that satisfy $\alpha + \beta = 1$.

In addition, the total cost of the selected computing resources subset $\mathcal{S}$ can be expressed as:

$$C(\mathcal{S}) = \sum_{s \in \mathcal{S} \subseteq \mathcal{K}} u_s \tag{13}$$

where the total cost should not exceed the user budget U, i.e., $C(\mathcal{S}) \leq U$.

Therefore, the computing resources selection of the task offloading problem in site-collaboration mode can be formulated as:

$$\textbf{P2-1} : \max_{\mathcal{S} \subseteq \mathcal{K}} Q(\mathcal{S}) \tag{14}$$

$$s.t. \quad C(\mathcal{S}) \leq U \tag{15}$$

Stage-2: Task Offloading Decisions and Computing Resources Allocations. As can be seen from above, the set of computing resources that can be used to offload computation tasks from mobile devices can be denoted as $\mathcal{V} = \{1, 2, ..., V = S + 1\}$, which includes the current MEC site and the external MEC sites in computing resources subset $\mathcal{S}$.

For a given set of $\mathcal{V}$, the task offloading decision of computation task n generated from mobile device m can be extended to $z_{m,n} \in \{0, 1, 2, ..., V\}$, where $z_{m,n} = 0$ indicates that the computation task n generated from mobile device m is executed at the mobile device m, $z_{m,n} = 1$ indicates that the computation task n generated from mobile device m is executed on the edge server in the current MEC site, and $z_{m,n} > 1$ indicates that the computation task n generated from mobile device m is transferred to external MEC sites in EdgeCPNs for execution. We suppose that the return delay of computation result is ignored. Thus, the total delay of task offloading in site-collaboration mode consists of mobile device execution delay, edge server transmission delay, edge server execution delay, external MEC site transmission delay, and external MEC site execution delay.

Similarly, the mobile device execution delay can be represented as:

$$T_m^{md} = \sum_{n \in \mathcal{L}^3} \frac{f_{m,n}}{c_m^{md}}; \mathcal{L}^3 = \{n | z_{m,n} = 0\} \tag{16}$$

The edge server transmission delay can be represented as:

$$T_m^{me} = \sum_{n \in \mathcal{L}^4} \frac{d_{m,n}}{R_m}; \mathcal{L}^4 = \{n | z_{m,n} > 0\} \tag{17}$$

Let c_v^{cn} denote the computing power of MEC site v, where $c_v^{cn} \in \{c^{es}, c_1^{ms}, c_2^{ms}, ..., c_s^{ms}\}$. Given computing resources allocation $w_{m,n,v} \in [0, c_v^{cn}]$, the edge server execution delay can be expressed by:

$$T_m^{es} = \max_{n \in \mathcal{L}^5} \frac{f_{m,n}}{w_{m,n,v}}; \mathcal{L}^5 = \{n | z_{m,n} = 1\}, v = 1 \tag{18}$$

The external MEC site transmission delay can be calculated as:

$$T_{m,s}^{em} = \sum_{n \in \mathcal{L}^6} \frac{d_{m,n}}{r_s}; \mathcal{L}^6 = \{n | z_{m,n} = s + 1\} \tag{19}$$

The external MEC site execution delay can be calculated by:

$$T_{m,s}^{ms} = \max_{n \in \mathcal{L}^6} \frac{f_{m,n}}{w_{m,n,s+1}}; \mathcal{L}^6 = \{n | z_{m,n} = s + 1\} \tag{20}$$

The delay of offloading computation task from the current MEC site to the external MEC site can be expressed as:

$$T_m^{ad} = \max_{s \in \mathcal{S}} \{ \sum_{m=1}^{M} T_{m,s}^{em} + T_{m,s}^{ms} \} \tag{21}$$

Therefore, the total delay required to complete the computation tasks of mobile device m can be represented as:

$$T_m^{SC} = \max\{T_m^{md}, T_m^{me} + T_m^{es}, T_m^{me} + T_m^{ad}\} \tag{22}$$

The task offloading problem in the site-collaboration mode involves the optimization of the following decision variables: i) computing resources selection $s \in \mathcal{S} \subseteq \mathcal{K}$, is a binary variable; ii) task offloading decision $z_{m,n} \in \{0, 1, 2, ..., V\}$, is a integer variable; iii) computing resources allocation $w_{m,n,v} \in [0, c_v^{cn}]$, is a continuous variable. Thus, following **P2-1**, the objective of stage-2 of the task offloading problem in the site-collaboration mode is to minimize the total delay for all mobile devices, which can be defined as:

$$\textbf{P2-2} : \min_{z_{m,n}, w_{m,n,v}} \sum_{m=1}^{M} T_m^{SC} \tag{23}$$

$$s.t. \quad z_{m,n} \in \{0, 1, 2, ..., V = S + 1\}, \forall m, \forall n \tag{24}$$

$$w_{m,n,v} \in [0, c_v^{cn}], \forall m, \forall n, \forall v \tag{25}$$

$$T_m^{SC} \leq t_m, \forall m \tag{26}$$

$$\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} z_{m,n} w_{m,n,v} \leq c_v^{cn}, \forall v \tag{27}$$

IV. Two-stage Evolutionary Search

A. Overall Framework

We develop an efficient two-stage evolutionary search scheme to solve the task offloading problems in EdgeCPNs. The search for task offloading problem is decoupled into two stages, where stage-1 search the computing resources selection and stage-2 search the task offloading decisions and the computing resources allocations. Both stages employ the evolutionary computation [40], [41] methods to produce corresponding solutions.

The overall framework of the proposed scheme is shown in Fig. 4. In stage-1, the search process consists of four parts: initialization, evaluation, mutation and selection. To optimize

the computing resources selection, a computing resources subset is selected to maximize the total gain under user budget constraint. The output of the stage-1 can be used as the input for the stage-2. If the output of stage-1 for problem **P2-1** is an empty set, stage-2 focuses on solving problem **P1**, otherwise, problem **P2-2** is solved. In stage-2, the search process is composed of five parts: initialization, evaluation, crossover, mutation and selection. To optimize the task offloading decisions and computing resources allocations to minimize the total delay, the internal and external computing resources need to be jointly scheduled.

B. Computing Resources Selection

As shown in Algorithm 1, we propose a computing resources selection method to search the optimal computing resources subset. A computing resources subset $\mathcal{S} \subseteq \mathcal{K}$ can be represented by a boolean vector $\{0,1\}^K$, where 0^K denotes an empty set. Similar to [42], a surrogate objective function that simultaneously takes total gain and total cost into account is introduced, which can be expressed as:

$$H(\mathcal{S}) = \begin{cases} Q(\mathcal{S}), & S = 0 \\ Q(\mathcal{S})/(1 - e^{-\lambda_Q C(\mathcal{S})/U}), & S > 0 \end{cases} \tag{28}$$

where λ_Q is the submodularity ratio of the original objective function. A large value of the surrogate objective function indicates a smaller total cost and larger total gain.

During the evolutionary process, a population of individuals is maintained. Each individual represents a set of computing resources to be selected, where the computing resources labelled as 1 constitute a candidate computing resources subset. First, the population is initialized with a single individual, which represents an empty set. Then, the offspring individuals are produced via a bit-wise mutation operator. For a randomly selected individual of the population, the bit-wise mutation operator generates its offspring individual by flipping each bit of the selected individual independently with probability $1/K$, where K is the size of set $\mathcal{K}$. When a new individual is generated, the fitness of the individual is evaluated by calculating corresponding total cost, total gain, and surrogate. Note that it must guaranteed that the user budget constraint is satisfied at any time of the individual evaluation. The offspring individual is compared only with the existing individuals in

Algorithm 1 Computing resources selection

Input: the optional computing resources set $\mathcal{K}$, the user budget U, the objective function **P2-1**
Output: the computing resources subset $\mathcal{S}$

Initialize population $P = \{p^1\}$, where $p^1 = 0^K$;
Let $h^0 = q^0 = 0^K$, and $t = 0$;
while $t < T$ **do**
  Randomly Select a individual p^{rd} from P;
  Generate p' from p^{rd} via bit-wise mutation operator;
  if $C(p') \leq U$ **then**
    Let $L = |p'|$;
    if $cmp(L) = \emptyset$ **then**
      $P = P \cup \{p'\}$, $h^L = p'$, and $q^L = p'$;
    else
      if $H(p') \geq H(h^L)$ **then**
        $h^L = p'$;
      end if
      if $Q(p') \geq Q(q^L)$ **then**
        $q^L = p'$;
      end if
      Let $O = P \setminus cmp(L)$;
      Update population $P = O \cup \{h^L\} \cup \{q^L\}$;
    end if
  end if
  $t = t + 1$;
end while
return p^{best} with the maximum objective value;

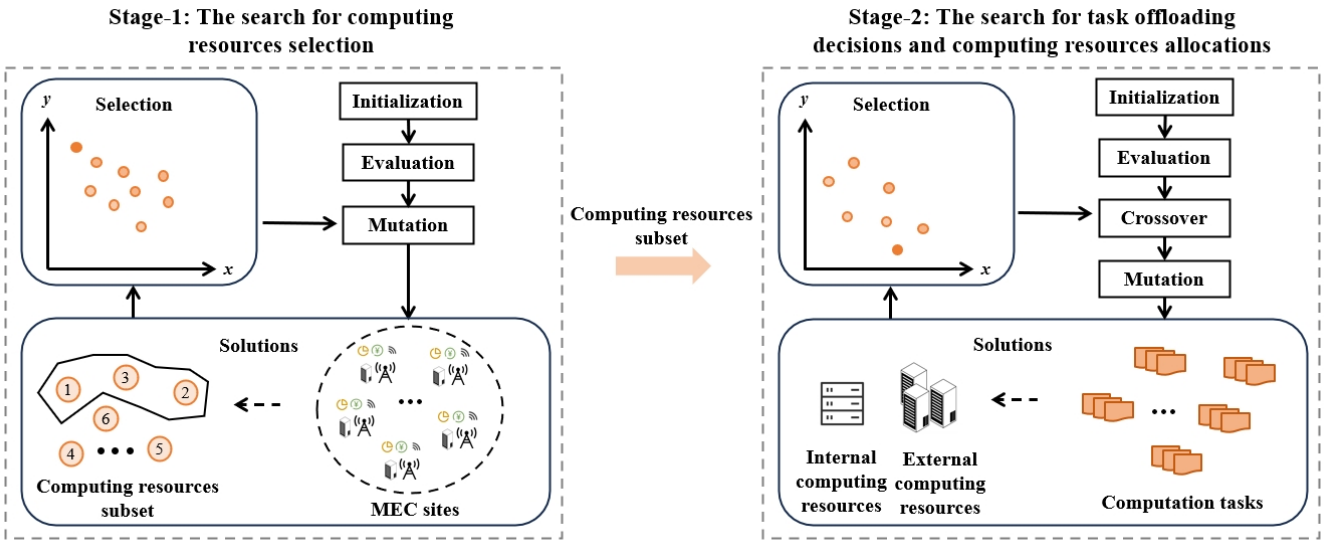

Fig. 4. Overall framework of the proposed scheme.

the population which have the same size as it. After several iterations, the individual with the maximum objective value of objective function **P2-1** is output as the computing resources set.

C. Task Offloading Decision and Computing Resources Allocations

As shown in Algorithm 2, we propose a task offloading decision and computing resources allocation method to optimize the joint use of internal and external computing resources. Each individual in the population represents a complete solution to the task offloading problem, which can be decoded into two parts, with half of the individual representing the task offloading decision vector and the another half representing the computing resources allocation vector. The best solution obtained from Algorithm 1 is used as one of the inputs to Algorithm 2 to reduce search space and improve search efficiency.

At the beginning of the evolutionary process, NP individuals are randomly initialized according to the computing resources subset $\mathcal{S}$ obtained in Algorithm 1. Then, the fitness of each individual is calculated during the individual evaluation. In each iteration, the worst and oldest individuals in the population are removed to reduce negative guidance. As their replacement, a randomly generated individual is added to the population to maintain diversity. To produce new individuals, we randomly select two individuals from the current population as parent individuals. By applying a binary crossover operator and a polynomial mutation operator [43], [44], two offspring individuals can be generated. All offspring individuals are added to the offspring population and evaluated. After that, we use a elite selection operator to select the individuals with good fitness from current population and offspring population to the next generation [45]. Ultimately, the individual with the minimum objective value is output as the task offloading decisions and computing resources allocations for computation tasks. The proposed two-stage evolutionary search scheme provides a flexible way to deal with the task offloading problems in EdgeCPNs. If the computing resources subset found in stage-1 is the empty set, the solution to problem **P1** can be obtained, otherwise the solution to problem **P2-2** can be obtained.

Algorithm 2 Task offloading Decision and computing resources allocation

Input: the computing resources subset $\mathcal{S}$, the population size NP, the objective function **P1** ($\mathcal{S} = \emptyset$), the objective function **P2-2** ($\mathcal{S} \neq \emptyset$)

Output: the task offloading decisions and the computing resources allocations for computation tasks

Randomly initialize population $P = \{p^1, p^2, ...p^{NP}\}$;
Let $t = 0$;
for $i = 1$ to NP **do**
  Evaluate individual p^i;
end for
while $t < T$ **do**
  $P = P - \{p^{worst}\} - \{p^{oldest}\} \cup \{p^{rd}\}$;
  Let $O = \emptyset$;
  for $i = 1$ to $NP/2$ **do**
    Randomly select two parent individuals p_1^p and p_2^p from P;
    Generate two offspring individuals p_1^c and p_2^c from p_1^p and p_2^p via simulated binary crossover and polynomial mutation operators;
    Evaluate offspring individuals p_1^c and p_2^c;
    $O = O \cup \{p_1^c, p_2^c\}$;
  end for
  Select NP individuals from $(O \cup P)$ to P via elite selection operator;
  $t = t + 1$
end while
return p^{best} with the minimum objective value;

D. Complexity Analysis

For the proposed two-stage evolutionary search scheme, the objective of computing resources selection in stage-1 is to identify a computing resources subset that has maximized total gain and whose total cost is no greater than the user budget. This problem has been proven to be NP-Hard. Within each iteration of Algorithm 1, we randomly select an individual p^{rd} from the current population whose offspring individual p' is generated via a bit-wise mutation operator. Since the individual has dimension K, the complexity of applying the mutation operation to it can be denoted as $O(K)$. The total number of iterations of Algorithm 1 is T, the time complexity of computing resources selection in stage-1 is $O(T * K)$.

Based on the computing resources subset identified in stage-1, the task offloading decision and the computing resources allocation are jointly optimized to minimize the total delay in stage-2, and also NP-Hard. In the initialization step, NP individuals of dimension $M * N$ are evaluated, the complexity of the evaluation operation can be denoted as $O[NP * M * N]$. In the main cycle, individuals in the current population undergo crossover and mutation operations to produce offspring individuals. The binary crossover operator is used to generate NP offspring individuals of dimension $M * N$ with a complexity of $O(NP * M * N)$. Similarly, the polynomial mutation operator has a certain probability of being applied to offspring individuals, its complexity can be denoted as $O(NP * M * N)$. After that, NP offspring individuals of dimension $M * N$ are evaluated with a complexity of $O(NP * M * N)$. The complexity of selecting NP individuals from the current population to the next generation is denoted as $O(NP)$. The total number of iterations of Algorithm 2 is T, the time complexity of task offloading decision and computing resources allocation in stage-2 is $O(T * NP * M * N)$.

V. Numerical Results

A. Setup

Based on the system model and problem formulation introduced in Section III, we set up the computing power of each mobile device to be uniformly distributed between

$[0.5 * 10^9, 1.5 * 10^9]$ GHz. The transmission power of each mobile device is set to 300 mW. The bandwidth allocated to each mobile device is set to 2 MHz. The noise power is -75 dBm. The input data size and the required CPU cycles of each computation task are randomly generated between [400, 1000] KB and $[0.1 * 10^9, 1.0 * 10^9]$ cycles. An edge server with computing power of 20 GHz is equipped in the current MEC site. In addition, we set up 100 external MEC site in EdgeCPNs. The available computing resources and the rental cost of each of the external MEC site are randomly distributed in the range of [5, 15] GHz and [10, 100]. The transmission rates from the current MEC site to external MEC sites are randomly set to [10, 20] Mbps.

To evaluate the effectiveness of the proposed two-stage evolutionary search scheme, the comparison methods are shown below:

- **CTMD:** For all $m \in \mathcal{M}$ and $n \in \mathcal{N}_m$, the task offloading decision $x_{m,n}$ is set to 0. This indicates that all computation tasks are executed at the mobile devices.
- **CTES:** For all $m \in \mathcal{M}$ and $n \in \mathcal{N}_m$, the task offloading decision $x_{m,n}$ is set to 1. This indicates that all computation tasks are executed at the edge server in the current MEC site.
- **CTRD:** For all $m \in \mathcal{M}$ and $n \in \mathcal{N}_m$, the task offloading decision $x_{m,n}$ is randomly set to 1 or 0. This indicates that the computation tasks can be randomly executed at the mobile devices or on the edge server in the current MEC site.
- **TESA-P1:** The computing resources subset $\mathcal{S}$ is an empty set. This suggests that the computation tasks offloaded from mobile devices are processed through site-autonomy mode, which schedules computing resources within the current MEC site to solve task offloading problem **P1**.
- **TESA-P2:** The computing resources subset $\mathcal{S}$ searched in the stage-1 is not the empty set. This suggests that the computation tasks offloaded from mobile devices are processed through site-collaboration mode, in which computing resources within the current MEC site and from the computing resources subset are jointly scheduled to solve task offloading problem **P2**.
- **SSES:** Without considering the process of computing resource selection, it uses a single-stage search scheme to optimize task offloading decisions and computing resource allocations for the task offloading problem in EdgeCPNs.
- **RDMS:** Similar to the proposed method, it is a two-stage search scheme. The difference is that it randomly identifies a computing resources subset in the the process of computing resources selection.
- **GCCA:** A meta-heuristic algorithm using a single-stage search scheme [46], which is designed to solve the task offloading problem in resource-constrained environments.

B. Evaluation of Task Offloading

Fig. 5 presents the total delay obtained by the four comparison methods under different numbers of mobile devices. It shows that TESA-P1 method outperforms the other methods in all problem instances. For the CTRD method, due to the irrational utilization of the computing resources, which results in the worst delay performance. In most cases, the CTES method achieves better delay performance than the CTMD method by offloading computation tasks to the edge server within the current MEC site. However, the delay performance of the CTES method degrades rapidly as the size of the problem instance increases. This means that overloaded computation tasks on the edge server can lead to intense competition for computing resources. Fig. 6 shows the total delay achieved by the four comparison methods for different numbers of computation tasks. Although the delay performance of the TESA-P1 method is better than both the CTMD and CTES methods, its advantages are not very obvious. As the number of computation tasks increases, the execution delay of the mobile device increases. This will force more computation tasks to be uploaded to the edge server for execution. Since the computing resources of the edge server within the current MEC site are non-expandable, the delay performance of the TESA-P1 method is limited.

When the user budget is given, the GCCA method, the SSES method, the RDMS method, and the TESA-P2 method are able to provide additional computing resources for task offloading. Fig. 7 presents the total delay versus the number of mobile devices. When the available computing resources increases, it can result in additional transmission delay by offloading computation tasks to external MEC sites. Meanwhile, the increase in computing resources can also reduce the execution delay of computation tasks. The TESA-P2 achieves the best delay performance by providing on-demand scheduling of computing resources. Its advantage is that it can select the most appropriate MEC sites from the network for collaborative task offloading. The delay performance of RDMS method is almost the same as that of SSES method when the instance with a large number of mobile devices. This indicates that the computing resources selection is what largely influences the effectiveness of task offloading. Meanwhile, the GCCA method shows better search efficiency compared to SSES and RDMS methods. Fig. 8 shows the total delay versus the number of computation tasks. It can be seen that the total delay of the SSES method and the GCCA method gradually outperform the RDMS method when the number of computation tasks increases. Because the RDMS method performs a random search for computing resources selection of the task offloading problem, it leads to a less efficient computing resources scheduling performance.

C. Analysis of Computing Resources Scheduling

Different computing resources scheduling performance can be obtained when different user demands are set. Fig. 9 shows the total delay obtained by the four comparison methods under different user budgets. It can be observed that the delay performance of TESA-P2 method is significantly better than that of GCCA, SSES and RDMS methods. This demonstrates that the TESA-P2 method has excellent performance in scheduling computing resources from the network. On the contrary, the SSES method performs a single-stage search for the task

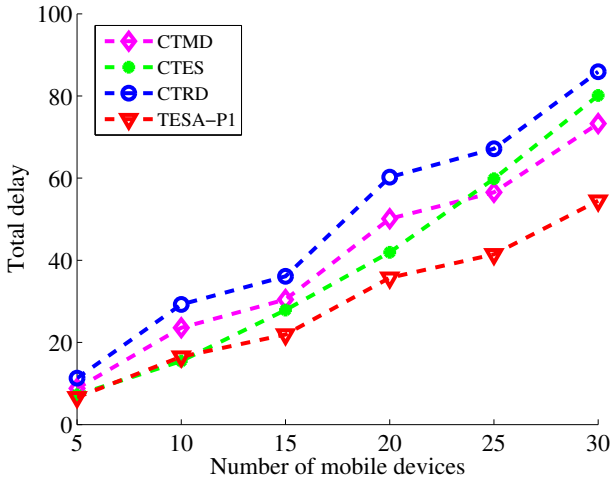

Fig. 5. Total delay of **P1** under different numbers of mobile devices.

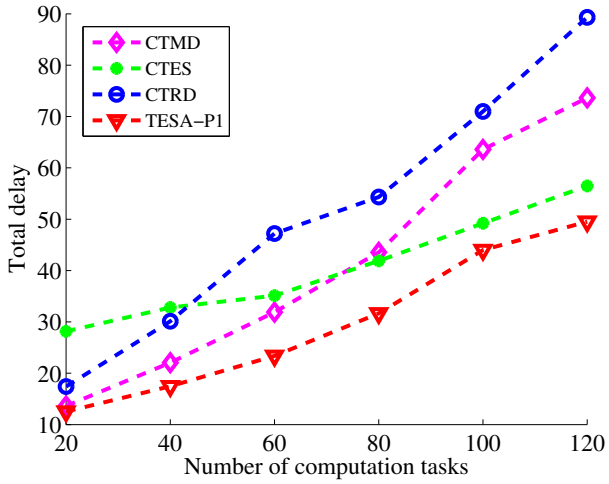

Fig. 6. Total delay of **P1** under different numbers of computation tasks.

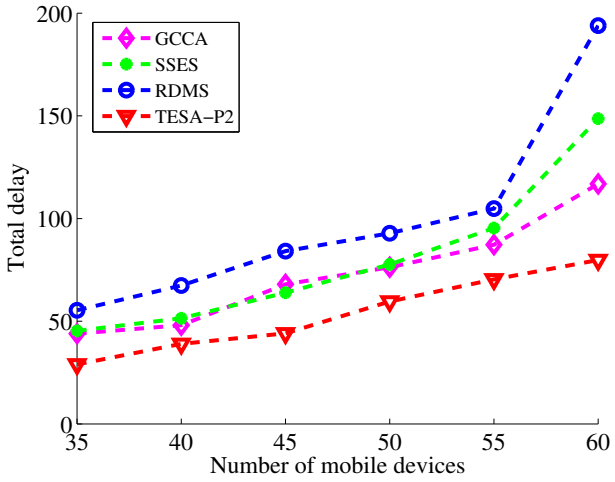

Fig. 7. Total delay of **P2** under different numbers of mobile devices.

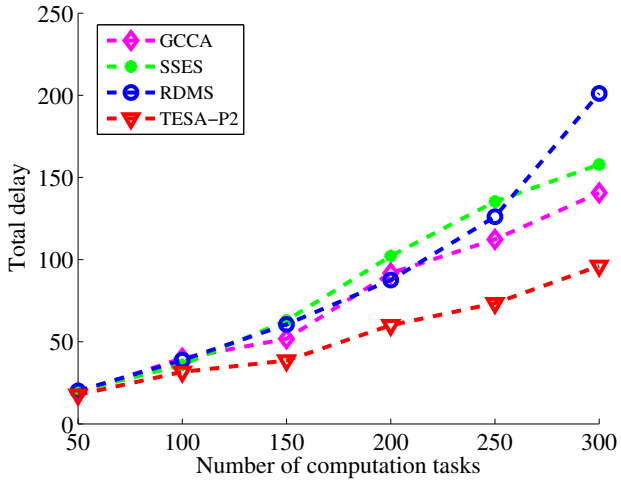

Fig. 8. Total delay of **P2** under different numbers of computation tasks.

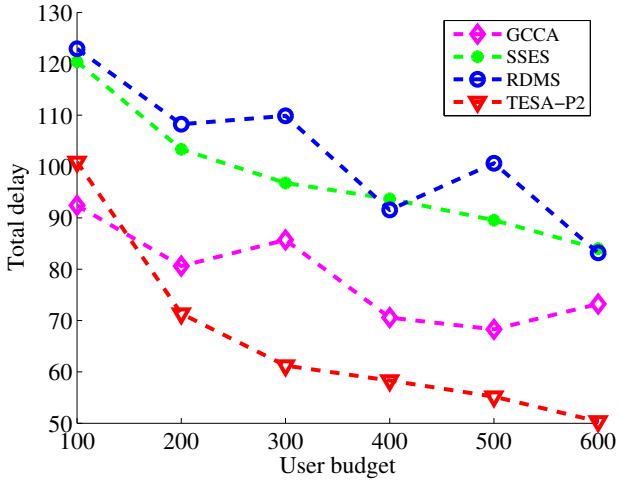

Fig. 9. Total delay with different user budgets.

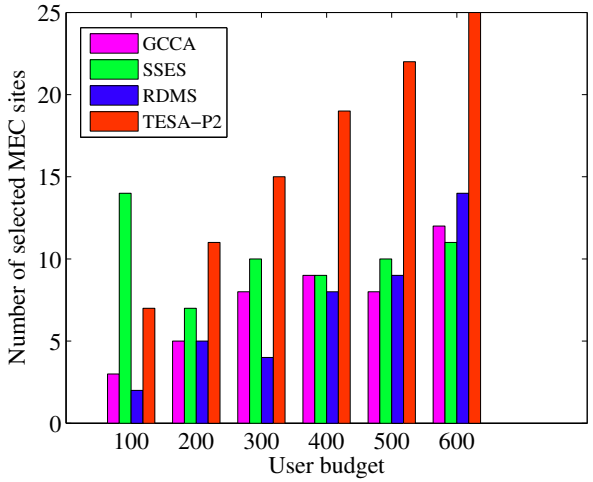

Fig. 10. Size of computing resources subset with different user budgets.

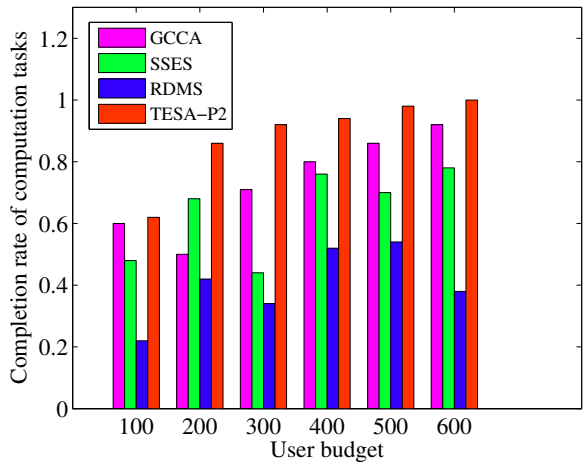

Fig. 11. Completion rate of computation tasks with different user budgets.

offloading problems in EdgeCPNs, which yields inefficient scheduling performance for computing resources. The RDMS method suffers from an unstable scheduling performance of computing resources as it fails to find the most appropriate computing resources subset in the search of the stage-1. The GCCA method performs better than the SSES method and the RDMS method, but with instability.

Fig. 10 shows the number of selected MEC sites obtained by the four comparison methods under different user budgets and Fig. 11 gives the corresponding completion rate of computation tasks. In the first case, the SSES method selected more MEC sites than the GCCA method, yet obtained a lower completion rate of computation tasks. This indicates that the SSES methods do not make sufficient utilization of the fragmented computing resources in EdgeCPNs. Although the TESA-P2 method selects fewer MEC sites than the SSES method, it achieves a more efficient computing resources scheduling performance based on the computing resources subset found in stage-1. In addition, the TESA-P2 method showed the best completion rate of computation tasks in all cases. The RDMS method randomly selects a subset of computing resources in stage-1, whose completion rate of computation tasks is limited by the selection of inappropriate computing resources.

Fig. 12 presents the total delay obtained by the four comparison methods under different numbers of external MEC sites in EdgeCPNs. We observe that the delay performances of the SSES and RDMS methods are instability. When the number of external MEC sites in EdgeCPNs increases, the search space for the task offloading problem becomes larger. For the SSES method, it is difficult to obtain efficient task offloading decisions and computing resources allocations for computation tasks from a large-scale search space based on a single-stage search. Although the GCCA method also suffers from instability, its delay performance is significantly better than that of the SSES and RDMS methods. In the TESA-P2 method, by decoupling the task offloading problems in EdgeCPNs into two stages, the search for computing resources selection in stage-1 can reduce the difficulty of searching for task offloading decisions and computing resources allocations in stage-2. For the RDMS method, it is not able to obtain an appropriate computing resources subset in the search for computing resources selection in stage-1. The TESA-P2 method can greatly improve the computing resources scheduling performance by considering the total cost and total gain of the selected computing resources subset. Fig. 13 shows the convergence trend obtained by the four comparison methods during the evolutionary process. It can be seen that the TESA-P2 method obtains better convergence performance than the GCCA, SSES, and the RDMS methods. This again validates the importance in solving the computing resources selection of task offloading problems and the effectiveness of the proposed two-stage evolutionary search scheme.

VI. Conclution

In this paper, we introduced the concept of edge computing power network (EdgeCPN) to efficiently utilize fragmented

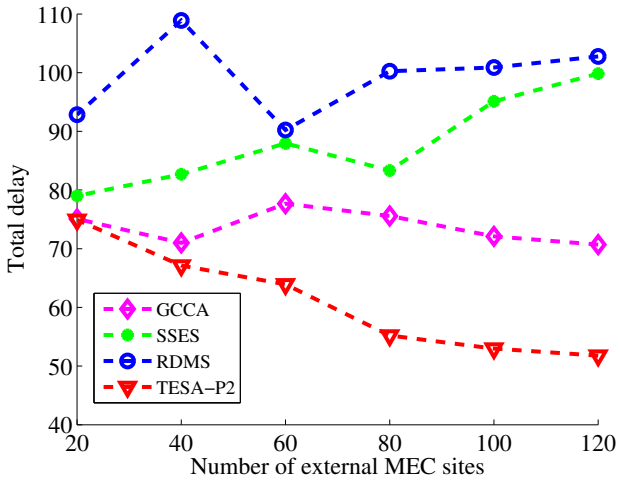

Fig. 12. Total delay under different numbers of external MEC sites.

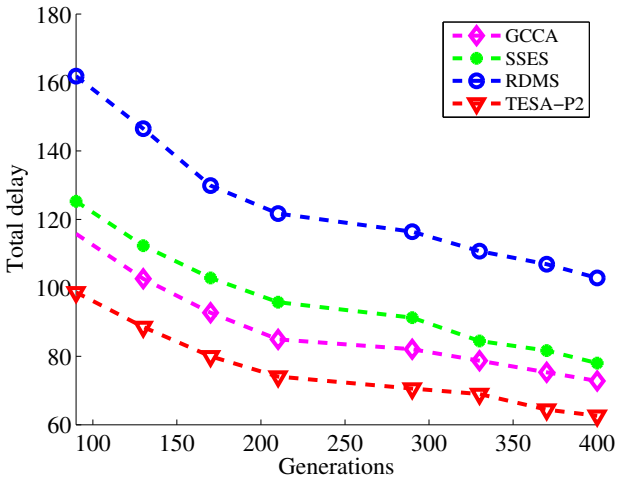

Fig. 13. Convergence trend of different comparison algorithms.

computing resources across different MEC sites in CPNs. For the task offloading problems in EdgeCPNs, in addition to optimizing the task offloading decisions and computing resources allocations, we also considered the optimization for computing resources selection. To this end, we designed an on-demand computing resources scheduling model for task offloading in EdgeCPNs to improve the performance of computing resources scheduling. Building on this foundation, we decouple the search for task offloading problems in EdgeCPNs into two stages, where stage-1 solves computing resources selection by searching a computing resources subset with the highest gain under user budget constraint, and stage-2 concurrently solves task offloading decisions and computing resources allocations according to the computing resources subset found in stage-1.

Our scheme is applicable not only to existing MEC network scenarios but also to emerging CPN scenarios. In particular, computing resources selection can be regarded as a general strategy for task offloading in CPNs. In the future, computing resources are organically associated with network resources. This enables users to weigh the performance and cost to determine the appropriate computing resources from a huge pool of computing resources to meet their diverse needs.

REFERENCES

[1] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, pp. 591–624, 2023.

[2] S. Duan, F. Lyu, H. Wu, W. Chen, H. Lu, Z. Dong, and X. Shen, "Moto: Mobility-aware online task offloading with adaptive load balancing in small-cell mec," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 645–659, 2024.

[3] Z. Gao, L. Yang, and Y. Dai, "Large-scale cooperative task offloading and resource allocation in heterogeneous mec systems via multiagent reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2303–2321, 2024.

[4] F. Lyu, J. Ren, N. Cheng, P. Yang, M. Li, Y. Zhang, and X. S. Shen, "Lead: Large-scale edge cache deployment based on spatio-temporal wifi traffic statistics," *IEEE Transactions on Mobile Computing*, vol. 20, no. 8, pp. 2607–2623, 2021.

[5] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 393–413, 2014.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[7] Y. Ren, S. Shen, Y. Ju, X. Wang, W. Wang, and V. C. Leung, "Edgematrix: A resources redefined edge-cloud system for prioritized services," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 610–619.

[8] C. Zhang, C. Hu, T. Wu, L. Zhu, and X. Liu, "Achieving efficient and privacy-preserving neural network training and prediction in cloud environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4245–4257, 2023.

[9] C. Hu, C. Zhang, D. Lei, T. Wu, X. Liu, and L. Zhu, "Achieving privacy-preserving and verifiable support vector machine training in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3476–3491, 2023.

[10] C. Yang, S. Lan, L. Wang, W. Shen, and G. G. Q. Huang, "Big data driven edge-cloud collaboration architecture for cloud manufacturing: A software defined perspective," *IEEE Access*, vol. 8, pp. 45 938–45 950, 2020.

[11] C. Yang, S. Lan, Z. Zhao, M. Zhang, W. Wu, and G. Q. Huang, "Edge-cloud blockchain and ioe-enabled quality management platform for perishable supply chain logistics," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3264–3275, 2023.

[12] C. Yang, Q. Chen, Z. Zhu, Z.-A. Huang, S. Lan, and L. Zhu, "Evolutionary multitasking for costly task offloading in mobile edge computing networks," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[13] S. Bebortta, D. Senapati, C. R. Panigrahi, and B. Pati, "Adaptive performance modeling framework for qos-aware offloading in mec-based iiot systems," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 10 162–10 171, 2022.

[14] Y. Ye, L. Shi, X. Chu, R. Q. Hu, and G. Lu, "Resource allocation in backscatter-assisted wireless powered mec networks with limited mec computation capacity," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10 678–10 694, 2022.

[15] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, 2022.

[16] C. Zhang, X. Luo, J. Liang, X. Liu, L. Zhu, and S. Guo, "Pota: Privacy-preserving online multi-task assignment with path planning," *IEEE Transactions on Mobile Computing*, pp. 1–13, 2023.

[17] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. 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.

[18] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 11, pp. 3836–3851, 2022.

[19] P. Wang, K. Li, B. Xiao, and K. Li, "Multiobjective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 11 737–11 748, 2022.

[20] Z. Wang, Z. Zhou, H. Zhang, G. Zhang, H. Ding, and A. Farouk, "Ai-based cloud-edge-device collaboration in 6g space-air-ground integrated power iot," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 16–23, 2022.

[21] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[22] S. Long, Y. Zhang, Q. Deng, T. Pei, J. Ouyang, and Z. Xia, "An efficient task offloading approach based on multi-objective evolutionary algorithm in cloud-edge collaborative environment," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 645–657, 2023.

[23] C. Yang, Y. Wang, S. Lan, L. Wang, W. Shen, and G. Q. Huang, "Cloud-edge-device collaboration mechanisms of deep learning models for smart robots in mass personalization," *Robotics Comput. Integr. Manuf.*, vol. 77, p. 102351, 2022.

[24] X. Tang, C. Cao, Y. Wang, S. Zhang, Y. Liu, M. Li, and T. He, "Computing power network: The architecture of convergence of computing and networking towards 6g requirement," *China Communications*, vol. 18, no. 2, pp. 175–185, 2021.

[25] Y. Zhang, C. Cao, X. Tang, R. Pang, S. Wang, and X. Wen, "Programmable service system based on sidaas in computing power network," in *2022 5th International Conference on Hot Information-Centric Networking (HotICN)*, 2022, pp. 67–71.

[26] Y. Lu, B. Ai, Z. Zhong, and Y. Zhang, "Energy-efficient task transfer in wireless computing power networks," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9353–9365, 2023.

[27] J. Li, H. Lv, B. Lei, and Y. Xie, "A computing power resource modeling approach for computing power network," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, 2022, pp. 1–2.

[28] Q. Shen, B.-J. Hu, and E. Xia, "Dependency-aware task offloading and service caching in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 12, pp. 13 182–13 197, 2022.

[29] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1320–1323, 2019.

[30] S. Wang, M. Chen, X. Liu, C. Yin, S. Cui, and H. Vincent Poor, "A machine learning approach for task and resource allocation in mobile-edge computing-based networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1358–1372, 2021.

[31] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[32] C. Liu, K. Liu, S. Guo, R. Xie, V. C. S. Lee, and S. H. Son, "Adaptive offloading for time-critical tasks in heterogeneous internet of vehicles," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7999–8011, 2020.

[33] M. Feng, M. Krunz, and W. Zhang, "Joint task partitioning and user association for latency minimization in mobile edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8108–8121, 2021.

[34] G. Peng, H. Wu, H. Wu, and K. Wolter, "Constrained multiobjective optimization for iot-enabled computation offloading in collaborative edge and cloud computing," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 723–13 736, 2021.

[35] Y. Laili, X. Wang, L. Zhang, and L. Ren, "Dsac-configured differential evolution for cloud-edge-device collaborative task scheduling," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2023.

[36] H. Yao, X. Duan, and Y. Fu, "A computing-aware routing protocol for computing force network," in *2022 International Conference on Service Science (ICSS)*, 2022, pp. 137–141.

[37] H. Hao, S. Yang, and W. Zhang, "Time-continuous computing task offloading mechanism for computing and network convergence," *Journal of Computer Research and Development*, vol. 60, no. 4, pp. 735–749, 2023.

[38] W. Sun, Z. Li, Q. Wang, and Y. Zhang, "Fedtar: Task and resource-aware federated learning for wireless computing power networks," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4257–4270, 2023.

[39] R. Zhou, X. Wu, H. Tan, and R. Zhang, "Two time-scale joint service caching and task offloading for uav-assisted mobile edge computing," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1189–1198.

[40] X. Li, G. Zhang, X. Zheng, and S. Hua, "Delay optimization based on improved differential evolutionary algorithm for task offloading in fog computing networks," in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, 2020, pp. 109–114.

[41] Z.-Y. Chai, Y.-J. Zhao, and Y.-L. Li, "Multi-task computation offloading based on evolutionary multi-objective optimization in industrial internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[42] Y. Sun, C. Lin, J. Ren, P. Wang, L. Wang, G. Wu, and Q. Zhang, "Subset selection for hybrid task scheduling with general cost constraints," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 790–799.

[43] J.-J. Lin, S.-C. Huang, and M.-K. Jiau, "An evolutionary multiobjective carpool algorithm using set-based operator based on simulated binary crossover," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3432–3442, 2019.

[44] K. Liagkouras and K. Metaxiotis, "An elitist polynomial mutation operator for improved performance of moeas in computer networks," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, 2013, pp. 1–5.

[45] J. Chen and Z. Xiao, "Research on adaptive genetic algorithm based on multi-population elite selection strategy," in *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)*, 2017, pp. 108–112.

[46] X. Chen, Y. Mao, H. Wang, Y. Xu, D. Li, S. Liu, and X. Zhao, "Data-driven task offloading method for resource-constrained terminals via unified resource model," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9703–9715, 2023.