# Evolutionary Multitasking for Costly Task Offloading in Mobile Edge Computing Networks

Chen Yang, *Member, IEEE*, Qunjian Chen, Zexuan Zhu, *Senior Member, IEEE*, Zhi-An Huang, Shulin Lan, Liehuang Zhu, *Member, IEEE*

***Abstract*—The offloading of computation-intensive tasks to an edge server close to the resource-constrained mobile devices can provide better application performance and user experience. However, with the rapid growth of mobile devices connected to the edge server, it is challenging to directly obtain an optimal task offloading scheme due to the increasing computational cost and problem scale. In this paper, we model the costly task offloading problem (CTOP) in mobile edge computing networks with the goal to achieve efficient joint optimization of energy consumption and processing latency for mobile devices. Inspired by the success of evolutionary multitasking in solving complex optimization problems by leverage the experience of simple optimization problems, we develop a novel multitasking framework whose effectiveness is demonstrated in solving CTOP. In this framework, auxiliary tasks are created for optimizing the local processing overhead and the edge processing overhead of task offloading. On this basis, we propose an effective multitask evolutionary algorithm that includes segmented knowledge transfer and auxiliary task update. Specially, source and extended decision variables are considered as different knowledge to be utilized, while the auxiliary tasks are allowed to be updated dynamically. Related knowledge learned from cheap and simple auxiliary tasks promote the evolutionary search for CTOP. The experimental results verify the effectiveness of knowledge transfer. Compared to other state-of-the-art multitasking algorithms as well as single-tasking algorithms, the proposed algorithm shows competitiveness in CTOP instances and can achieve better comprehensive performance in terms of energy consumption and processing latency.**

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 and 72201266, in part by the Fundamental Research Funds for the Central Universities under Grant E1E40805X2 and the State Key Laboratory of Digital Manufacturing Equipment and Technology under Grant DMETKF2021012. *(Corresponding author: Qunjian Chen and Shulin Lan.)*

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

Qunjian Chen is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: Chen-QJ@outlook.com).

Zexuan Zhu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: zhuzx@szu.edu.cn).

Zhi-An Huang is with the Center for Computer Science and Information Technology, City University of Hong Kong Dongguan Research Institute, Dongguan 523000, China (e-mail: huang.za@cityu.edu.cn).

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

***Index Terms*—Mobile edge computing, evolutionary multitasking, task offloading, auxiliary task.**

## I. Introduction

With the advances in mobile computing, Internet of Things and 5G/6G networks [1], the convergence of mobile technologies and artificial intelligence [2] facilitates mobile devices to support many resource-intensive smart applications like face recognition, image search and machine translation, etc. Offloading computation-intensive tasks of smart applications to a powerful edge server close to the resource-constrained mobile devices can provide better application performance and user experience. However, the rapid growth of mobile devices and computational tasks highlights ineluctable challenges of task offloading in terms of decision making and resource scheduling [3].

A prominent approach for the task offloading problem is that of evolutionary computation [4]. Song et al. [5] proposed an improved multi-objective evolutionary algorithm with two performance enhancing schemes, and considered the task-precedence constraints for the task offloading problem in mobile edge computing (MEC) networks. Pan et al. [6] presented a multi-objective clustering evolutionary algorithm for multi-workflow computation offloading, where an adaptive clustering approach is designed to improve the crossover operation. Although there have been many studies on task offloading, most of them cannot cope with the increasing computational cost and problem scale, and did not consider the issue of scalability. In [7] and [8], the costly task offloading problem (CTOP) is decomposed into multiple subproblems to simplify the search process by solving each subproblem separately. Although decomposition-based approaches can improve the search performance for CTOP, the subproblems are often solved independent and related knowledge learned is not exploited for more efficient optimization.

Evolutionary multitasking (EMT) is an emerging optimization paradigm in the field of evolutionary computation [9]. Contrary to traditional single-tasking evolutionary algorithms, EMT attempts to tackle multiple optimization problems simultaneously with a single evolutionary solver. Driving cross-domain transfer of knowledge between distinct but probably related optimization problems, the augmented implicit parallelism of population is achieved by the capacity of multitasking [10]. Growing efforts have been made to develop innovative EMT algorithms for performance gains in diverse studies [11].

The most recent studies indicate that one of the special characteristics of EMT is to take advantage of related and cheap auxiliary tasks to accelerate the evolutionary search of the costly optimization problem [12]-[14]. Compared to solving complex problems directly, the useful information gained from solving related and simple problems can help discover promising solutions and save the total evaluation cost. Even though existing EMT algorithms show strong competitiveness in dealing with costly optimization problems, key issues still exist to be investigated in terms of how to configure auxiliary tasks in a multitasking environment and how to efficiently exploit the useful information from the auxiliary tasks. Therefore, this paper proposes a novel EMT framework whose effectiveness is demonstrated in solving CTOP. Our method is to transform CTOP into a multi-task optimization problem by creating a set of highly correlated and computationally cheap auxiliary tasks, which aims to capture favorable knowledge from the auxiliary tasks to promote the search efficiency for CTOP. The auxiliary tasks are not only used for enhancing the capability of global search but also the capability of local search of the multitasking solver. The proposed method is expected to accelerate the convergence and improve the solution quality for CTOP. To the best of our knowledge, this is the first study to introduce EMT to solve CTOP. Specifically, the contributions of this article are given as follows.

1) We build a system model for the costly task offloading problems in MEC networks with the goal to achieve joint optimization of energy consumption and processing latency, in which task offloading decision, computing resource allocation and transmission resource allocation are involved. In addition, we create two types of auxiliary tasks with the aim of minimizing the edge processing overhead and local processing overhead of task offloading, respectively. The auxiliary tasks possess different similarities to the CTOP and can contribute to the efficient CTOP solving.

2) We develop a novel evolutionary multitasking framework, in which the CTOP is transformed into a multitasking optimization problem. The created auxiliary tasks are used to improve the evolutionary search of the CTOP by sharing favorable knowledge. The collaborative searching can greatly increase the efficiency and scalability of optimization algorithms in tackling the CTOP.

3) We propose an effective multitask evolutionary algorithm with segmented knowledge transfer and auxiliary task update to realize efficient CTOP solving using the above multitasking framework. Experimental results show that the knowledge transfer from auxiliary tasks to CTOP can accelerate the convergence and improve the scalability of the algorithm.

The rest of this paper is organized as follows. Section II provides a brief review of related work. We present the system model and construct the auxiliary tasks in Section III. In Section IV, the proposed method is introduced in detail. The experimental results and analyses are presented in Section V. Section VI concludes this paper and discuss future work.

## II. Background

### A. Multitask Optimization

The goal of multitask optimization (MTO) [1] is to provide a single optimizer that deals with multiple optimization tasks at the same time. Without loss of generality, given an optimization scenario in which $K$ optimization tasks are to be solved simultaneously. We assume that all optimization tasks are minimization problems and may have some equality and/or inequality constraints. The $j$-th optimization task is represented as $T_j$. Meanwhile, its search space is denoted as $X_j$, the corresponding objective function is defined as $F_j\colon X_j \rightarrow \mathbb{R}$. With this setting, the MTO paradigm can be formulated as follows:

$$\{x_1, x_2, \dots, x_K\} = argmin\{F_1(x_1), F_2(x_2), \dots, F_K(x_K)\} \quad (1)$$

where $x_j$ denotes a feasible solution in search space $X_j$. It is noteworthy that the search space $X_1, X_2, \dots, X_K$ of optimization task $T_1, T_2, \dots, T_K$ may be similar but often different. An effective way is to construct a unified search space [2], in which each of the component optimization tasks can be viewed as an additional influence factor to facilitate the knowledge transfer in a multitasking environment. In this paper, the CTOP is considered as the target optimization problem, while the auxiliary tasks are treated as other component optimization problems, they are allowed to be processed simultaneously in a multitasking environment.

### B. Task offloading

The task offloading is of great interest in MEC networks, which has been extensively studied in recent years. In general, the execution latency is usually one of the most concerned performance criteria. Dai et al. [15] proposed a coding-assisted multi-objective evolutionary algorithm to optimize the service delay and the network access cost, which considered the packet encoding and network interface selection. Wang et al. [16] constructed a distributed traffic management system, the average system response time is minimized by scheduling the message flow allocation among vehicle-based fog nodes. Ning et al. [17] presented a three-layer vehicular fog computing model to effectively offload network traffic, the integration of cloudlet and fog nodes is used to reduce the response delay. Chen et al. [18] considered the time-varying network dynamics in computation offloading for a virtual MEC system. They proposed two double DQN-based online learning algorithms to deal with the stochastic computation offloading problem with time-varying communication qualities and computation resources.

In real-world situations, the energy consumption is another important factor that must also be considered. In [19], Wang et al. realized the joint optimization of computational speed, transmit power and offloading ratio. They proposed a locally optimal algorithm with the univariate search technique to minimize the energy consumption and execution latency. An optimization framework that enables task offloading from a single mobile device to multiple edge devices is proposed in [20], which aims to minimize execution latency and energy consumption by coupling task allocation decisions and frequency scaling. Moreover, there are some studies focusing on the tradeoff between execution latency and energy consumption. To this end, Zhang et al. [21] proposed an energy-aware offloading scheme under the limited energy, which jointly optimizes communication and computation resource allocation in single and multicell MEC networks. Wang et al. [22] proposed an integrated framework to minimize the overall consumption of time and energy. The optimization problem consists of computation offloading decision, physical resource block allocation, and computation resource allocation.

However, most works ignored the impact of the explosive growth in the number of mobile devices and computation tasks, especially when dealing with large-scale or large computing task offloading problems, which results in unaccepted user experience and insufficient scalability. Although there are some studies that decomposed the target problem into several subproblems, the subproblems are only optimized individually rather than helping the solving of the target problem through knowledge sharing. Therefore, we conceive a novel task offloading mechanism that leverages the characteristics of evolutionary multitasking to take advantage of auxiliary tasks to accelerate the search for the CTOP.

### C. *Evolutionary Multitasking*

One trend in the field of optimization has been to effectively multitask. Recently, the concept of evolutionary multitasking indicates that the genetic material evolved for one problem may be effective for another as well. In [1], Gupta et al. proposed a multifactorial evolutionary algorithm (MFEA) that first demonstrated the superiority of evolutionary multitasking. They drew on the theory of multifactorial inheritance, where the genetic and cultural transmission enables MEFA to conduct a cross-domain search. As a rising research perspective, the effectiveness of MFEA has been verified in discrete, continuous, single-objective, as well as multi-objective optimization with some impressive results [3].

There are many researches concentrated on the knowledge transfer in the multitasking environment. Feng et al. [23] explored the implicit knowledge transfer under particle swarm optimization and differential evolution. Bail et al. [24] presented a linearized domain adaptation approach to alleviate the issue of negative knowledge transfer. Liang et al. [25] proposed a two-stage adaptive knowledge transfer algorithm based on population distribution, the individual search strategies are adjusted in different stages of evolution to improve convergence performance. In [26], the mapping matrix generated by subspace learning is introduced to transform the search space of the population, thereby reducing the negative knowledge transfer in the process of evolution. However, the component optimization problems in multitasking environment may have different computational complexity. When the resources are limited, more resources should be allocated to hard-to-handle component optimization problems. To achieve adaptive allocation of computational resources, Gong et al. [27] presented an online dynamic resource allocation strategy based on the evaluation of computational complexities of component optimization problems. For reducing negative transfer due to the increase in the total number of component optimization tasks, Xu et al. [28] proposed an adaptive EMT framework to adjust knowledge transfer frequency, knowledge source selection, and knowledge transfer intensity.

In particular, in the presence of correlation in component optimization problems, genetic transfer across domains usually leads to accelerated convergence of high-cost problems due to the rapid exploration of the search space by low-cost problems. Ding et al. [29] proposed a multitasking evolutionary optimization framework that uses decision variable translation and decision variable shuffling strategies to transfer knowledge from computationally inexpensive problems to help solve costly problems. Similarly, Zhang et al. [30] developed a surrogate-assisted multitasking method, which approximates individual fitness by building computationally inexpensive models to help reduce the number of function evaluations.

If the optimization scenario does not contain a simple problem, then it can be artificially constructed. To solve complex optimization problems, Ma et al. [13] constructed strongly related meme helper-tasks based on the problem structure or decision variable grouping, aiming to escape local optima and increase population diversity. Zhang et al [12]. created a band of small data proxies for the main task, the knowledge extraction and reuse from small data tasks lead to rapidly optimization of the large dataset. Qiao et al. [33] analyzed the similarity of the constrained Pareto front and the unconstrained Pareto front. The constrained multi-objective problem with dynamic constraint boundary is modeled as an auxiliary task solved simultaneously with the source problem using an optimization multitasking framework. Chen et al. [34] proposed an efficient computation resource allocation strategy for minions generated using subsampled small-data tasks, in which more resources are allocated for tasks with high correlation measures based on Bayes' rule.

Evolutionary multitasking provides a promising means to deal with the increasingly complex and costly optimization problems. However, many existing EMT algorithms cannot fully exploit the useful information in the candidate solutions corresponding to auxiliary tasks, leading to an implicit waste of resources. Our proposed algorithm includes a segmented knowledge transfer strategy, in which both source and extended decision variables (from promising candidate individuals in auxiliary tasks) in the unified search space can contribute different knowledge to significantly facilitate the search of target problem. Moreover, in many EMT algorithms, they tend to pre-place all the auxiliary tasks in the multitasking environment and transfer knowledge from different auxiliary tasks simultaneously. This may result in potential transfer conflicts (leading to slow convergence speed or negative transfer) in the target problem. Our proposed algorithm also includes an auxiliary task update mechanism, which allows the dynamic generation of different auxiliary tasks to maintain population diversity and generate new knowledge during the evolutionary process. There are two simultaneous auxiliary tasks in a multitasking environment, each pair exploiting different but complementary areas of the search space. This mechanism can reduce the probability of transfer conflicts and realize flexible and efficient local optimization.

## III. System Model and Auxiliary Tasks

### A. System Model

A typical 5G heterogeneous MEC network is composed of one macro base station (MBS) and $N$ small base stations (SBSs), as shown in Fig. 1. We consider that the MBS is equipped with an MEC server and has the powerful ability to process multiple computationally intensive tasks of smart applications concurrently. The MEC server provides computing and storage services for resource-constrained mobile users to execute their smart applications. For each SBS, mobile devices can initiate the request over the wireless link. The SBSs are overlaid by the MBS and connected to the MBS via wired link [10]. To support heterogeneous devices well, software-defined networking (SDN) [31] is introduced as an orchestrator that separates control functions from the data functions. These heterogeneous devices follow the scheduling of the SDN controller to transmit and process information.

For easy reference, Table I summarizes the key symbols used in this section.

We assume that there exits $N$ regions of mobile devices $M = \{M_1, M_2, \dots, M_N\}$ within the network. Mobile devices in each region are served by an SBS. In this network, the mobile device $i$ in region $j$ has a computation-intensive and non-decomposable task $T_{i,j} = \{c_{i,j}, d_{i,j}, t_{i,j}^{max}\}$ needs to be completed. Here $c_{i,j}$ is the total number of CPU cycles required to accomplish the computation task $T_{i,j}$, $d_{i,j}$ denotes the input data size of computation task $T_{i,j}$, $t_{i,j}^{max}$ represents the latency constraint of completing the computation task $T_{i,j}$. When processing a computation task, the mobile device can offload it to the MEC server with the consideration of time and energy consumption for the purpose of performance improvement. That means that computation tasks can be locally processed on a mobile device or on the MBS via task offloading. We define $O_{i,j} \in \{0,1\}$ to indicate the computation task processing mode, in which $O_{i,j} = 1$ represents the local processing and $O_{i,j} = 0$ for the edge processing.

1) *Local processing*

We assume that mobile devices have different computation capability. The CPU frequency of mobile devices is denoted as $F_{i,j}^{local}$, which is elastic that can be scheduled under the given constraints. In such a setting, we can obtain the local processing time $t_{i,j}^{local}$ as follows.

$$t_{i,j}^{local} = \frac{c_{i,j}}{F_{i,j}^{local}} \tag{2}$$

The corresponding energy consumption of a mobile device can be calculated as

$$e_{i,j}^{local} = z(F_{i,j}^{local})^2 c_{i,j} \tag{3}$$

where $z$ is an energy coefficient depending on the chip architecture [21].

There is a weight factor $w_{i,j}$ is used to reflect the tradeoff between processing time and energy consumption, which allows different weights are selected to meet the demands of mobile users in decision making. The overhead of the computation task processed locally can be expressed as

$$Q_{i,j}^{local} = w_{i,j} t_{i,j}^{local} + (1 - w_{i,j}) e_{i,j}^{local} \tag{4}$$

TABLE I
Key symbols and definitions

| Symbol | Definition |
|---|---|
| $T_{i,j}$ | the computation task |
| $c_{i,j}$ | the total number of CPU cycles required to accomplish the computation task |
| $d_{i,j}$ | the input data size of computation task |
| $t_{i,j}^{max}$ | the latency constraint of completing the computation task |
| $O_{i,j}$ | process computation task locally or at the edge |
| $F_{i,j}^{local}$ | the CPU frequency of mobile device |
| $t_{i,j}^{local}$ | the local processing time |
| $z$ | the energy coefficient depending on the chip architecture |
| $e_{i,j}^{local}$ | the local energy consumption |
| $w_{i,j}$ | the weight factor |
| $Q_{i,j}^{local}$ | the overhead of computation task processed locally |
| $B_{i,j}$ | the uplink bandwidth |
| $P_{i,j}$ | the transmission power |
| $H_{i,j}$ | the channel gain |
| $\sigma^2$ | the noise power |
| $r_{i,j}$ | the uplink rate |
| $t_{i,j}^{edge}$ | the total latency of edge processing |
| $F_{i,j}^{MEC}$ | the CPU frequency of MEC server |
| $e_{i,j}^{edge}$ | the energy consumption of data transmission |
| $Q_{i,j}^{edge}$ | the overhead of the computation task processed at edge |
| $e_{i,j}^{res}$ | the residual energy of mobile device |
| $F_{max}^{local}$ | the maximum CPU frequency of mobile device |
| $P_{max}$ | the maximum transmission power of mobile device |

Notably, the formulations in (2) and (3) allow the CPU frequency of mobile devices to exert influences on both processing time and energy consumption. Therefore, we consider to schedule the CPU frequency of mobile devices to achieve optimal task processing performance, this can be supported by using dynamic voltage and frequency scaling techniques [19].

2) *Edge processing*

We consider that each mobile user can initiate a request to the MEC server to offload the computationally intensive tasks, when the local resources of a mobile device cannot satisfy the need of the mobile user. The input data can be uploaded to the MEC server via the SBS, and the upload expenditure between SBS and MEC sever is ignored [22]. The uplink rate $r_{i,j}$ between mobile device and the SBS is calculated as

$$r_{i,j} = B_{i,j} log_2\left(1 + \frac{P_{i,j} H_{i,j}}{\sum_{n=1,n\neq j}^{N} \sum_{m=1}^{M_n} P_{m,n} H_{m,n+} \sigma^2}\right) \tag{5}$$

where $B_{i,j}$ is the uplink bandwidth and $P_{i,j}$ is the transmission power, respectively. Moreover, $H_{i,j}$ denotes the channel gain and $\sigma^2$ is the noise power. Note that the data transmission of a mobile device often suffers from interferences caused by other mobile devices in nearby regions.

Let $F_{i,j}^{MEC}$ be the CPU frequency of MEC server. We assume that the MEC server will process the computation task with maximum CPU frequency. Then the total latency of edge processing involves data transmission time and edge computing time can be given by

$$t_{i,j}^{edge} = \frac{d_{i,j}}{r_{i,j}} + \frac{c_{i,j}}{F_{i,j}^{MEC}} \tag{6}$$

Meanwhile, the corresponding energy consumption of mobile device can be calculated as

$$e_{i,j}^{edge} = \frac{d_{i,j}}{r_{i,j}} P_{i,j} \tag{7}$$

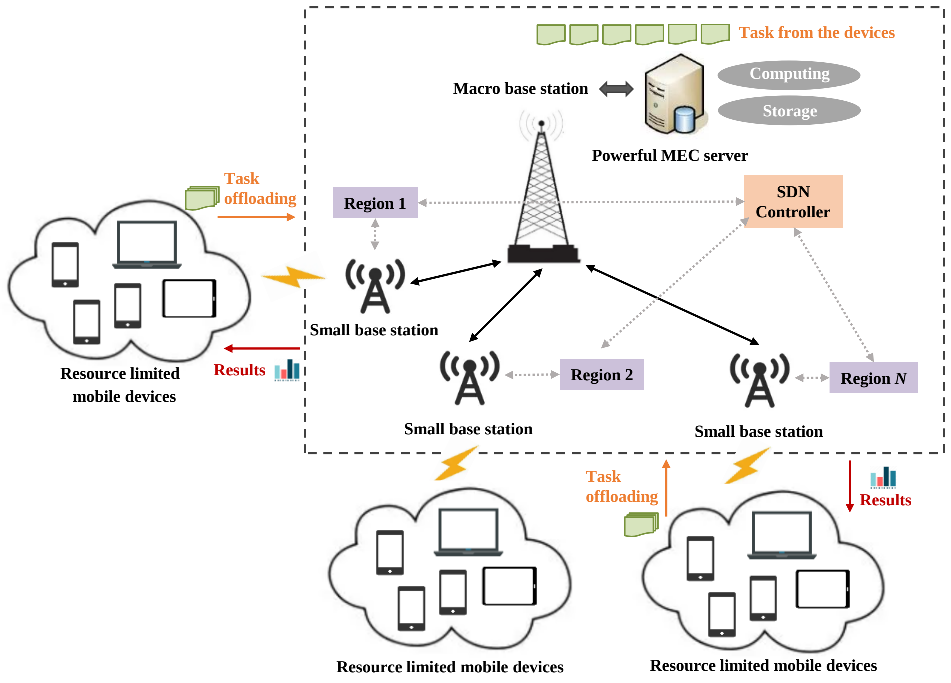


Fig. 1. The system model for task offloading.

Similarly, the overhead of the computation task processed at edge can be expressed as

$$Q_{i,j}^{edge} = w_{i,j}t_{i,j}^{edge} + (1-w_{i,j})e_{i,j}^{edge} \tag{8}$$

For simplicity, the consumption of time and energy of the mobile devices receiving the results from MEC server is ignored [31], because the size of returned results is usually much smaller than the input data size. In addition, the formulations in (6) and (7) indicate the transmission power $P_{i,j}$ has an impact on both time and energy consumption as well. By adjusting the transmission power, energy saving or delay reduction can be achieved.

3) *Problem formulation*

As previously described, we formulate the system model as an CTOP that optimizes the tradeoff between processing time of computation tasks and energy consumption of mobile devices in an MEC network. The total cost minimization problem with regard to the task offloading is shown as follows.

$$\begin{aligned} CTOP\colon \min_{O,F,P} \sum_{i=1}^{M_j}\sum_{j=1}^{N}\Bigg\{O_{i,j}\Bigg[w_{i,j}\left(\frac{d_{i,j}}{r_{i,j}}+\frac{c_{i,j}}{F_{i,j}^{MEC}}\right) \\ +\left(1-w_{i,j}\right)\frac{d_{i,j}}{r_{i,j}}P_{i,j}\Bigg]\Bigg\} \\ +\sum_{i=1}^{M_j}\sum_{j=1}^{N}\Bigg\{(1-O_{i,j})\Bigg[w_{i,j}\frac{c_{i,j}}{F_{i,j}^{local}} \\ +\left(1-w_{i,j}\right)z(F_{i,j}^{local})^2c_{i,j}\Bigg]\Bigg\} \end{aligned} \tag{9}$$

s.t.

$$C1\colon O_{i,j}\ \left(\frac{d_{i,j}}{r_{i,j}}+\frac{c_{i,j}}{F_{i,j}^{MEC}}\right)\ +(1-O_{i,j})\frac{c_{i,j}}{F_{i,j}^{local}} \le t_{i,j}^{max} \tag{9a}$$

$$C2\colon O_{i,j}\frac{d_{i,j}}{r_{i,j}}P_{i,j} + (1-O_{i,j})\ z(F_{i,j}^{local})^2c_{i,j} \le e_{i,j}^{res} \tag{9b}$$

$$C3\colon 0 < F_{i,j}^{local} \le F_{max}^{local} \tag{9c}$$

$$C4\colon 0 < P_{i,j} \le P_{max} \tag{9d}$$

$$C5\colon O_{i,j} \in \{0,1\} \tag{9e}$$

where $F_{max}^{local}$ and $P_{max}$ denote the maximum CPU frequency and maximum transmission power of mobile devices, respectively. The constraint $C1$ guarantees the total processing time of a computation task cannot exceed the maximum acceptable latency; constraint $C2$ indicates that the total energy consumption of a mobile device is less than its residual energy; constraints $C3$ and $C4$ restricts the CPU frequency and transmission power of mobile devices; $C5$ reflects that there are only two processing modes for computation tasks.

The CTOP formulated in this subsection is non-convex and NP-hard. The evolutionary algorithm, is a good choice to solve the problem. In consideration of the challenge in search complexity, we conceive to construct a set of related and easy-to-process auxiliary tasks to help solve the CTOP through evolutionary multitasking.

## B. *Auxiliary Tasks*

For the formulated CTOP, it is hard to obtain a high-quality solution directly. This suggests learning from related and cheap auxiliary tasks to reduce the difficulty of handling expensive and complex target problems. Currently, it is still an open issue about how to construct the auxiliary tasks. In [14], the auxiliary task has the same objective function as the constrained multi-objective problem, but with different constraints. The paper [12] used small data proxies to the main target task as the auxiliary tasks to quickly optimize for the target large dataset. In article [13], the strongly related meme helper-tasks are constructed through a multiobjectivization of the original task.

In this subsection, we attempt to create auxiliary tasks to perform quick search, which aims to improve the search efficiency for the CTOP through evolutionary multitasking. To this end, the auxiliary tasks are constructed to be highly correlated to the target problem in order to exploit the potential synergy between them with regard to the fitness landscape. In multitasking environments, auxiliary tasks are expected to provide additional favorable influences to rapidly advance the evolutionary search of the target problem at a relatively inexpensive computational cost.

The formulated CTOP includes the searching of task offloading decision variables, computing resource allocation variables, and transmission resource allocation variables. Moreover, it can be seen that the objective function of CTOP consists of two parts. The former component is the overhead of edge processing while the latter is the overhead of the local processing. Note that the task offloading decision can be potentially obtained through the analysis of edge processing overhead and local processing overhead. Therefore, we constructed two types of auxiliary tasks that are formally subproblems of the target problem. The motivation for constructing them is to find promising resource scheduling schemes for the target problem.

To reduce the local processing overhead of computation tasks, the first type of auxiliary tasks that finds a scheduling scheme for the CPU frequency of mobile devices, with the goal of minimizing the time and energy consumption of mobile devices, can be formed as

$$CTOP-F: \min_{F} \sum_{l=1}^{M_k'} [w_{l,k} \frac{c_{l,k}}{F_{l,k}^{local}} + (1-w_{l,k}) z\left(F_{l,k}^{local}\right)^2 c_{l,k}] \tag{10}$$

s.t.

$$C1: k \in \{1,2,\dots,N\}, M_k' \leq M_k \tag{10a}$$

$$C2: \frac{c_{l,k}}{F_{l,k}^{local}} \leq t_{l,k}^{max} \tag{10b}$$

$$C3: z(F_{i,j}^{local})^2 c_{l,k} \leq e_{l,k}^{res} \tag{10c}$$

$$C4: 0 < F_{l,k}^{local} \leq F_{max}^{local} \tag{10d}$$

where $k$ specifies the search region, $M_k'$ is used to control the number of mobile devices. In a more general case, the CTOP-F can be further simplified depending on the complexity and scale of the target problem, especially when a large number of mobile devices are connected to the MEC network. If a good solution is found for the CTOP-F, then it is likely to be beneficial for the target problem as well, since the CTOP-F is given a fitness landscape that is positively correlated with the target problem. Moreover, the CTOP-F is regarded as a simple optimization problem that requires significantly lower computation cost than the target problem. In a multitasking environment, the CTOP-F is expected to obtain promising solutions to assist the efficient optimization of the computationally expensive target problem, through knowledge transfer from CTOP-F to the target problem.

Another type of auxiliary tasks with the goal to minimize the edge processing overhead of computation tasks by scheduling the transmission power of mobile devices, can be represented as

$$CTOP-P: \min_{P} \sum_{l=1}^{M_k'} [w_{l,k} \left(\frac{d_{l,k}}{r_{l,k}} + \frac{c_{l,k}}{F_{l,k}^{MEC}}\right) + \left(1-w_{l,k}\right) \frac{d_{l,k}}{r_{l,k}} P_{l,k}] \tag{11}$$

s.t.

$$C1: k \in \{1,2,\dots,N\}, M_k' \leq M_k \tag{11a}$$

$$C2: \frac{d_{l,k}}{r_{l,k}} + \frac{c_{l,k}}{F_{l,k}^{MEC}} \leq t_{l,k}^{max} \tag{11b}$$

$$C3: \frac{d_{l,k}}{r_{l,k}} P_{l,k} \leq e_{l,k}^{res} \tag{11c}$$

$$C4: 0 < P_{l,k} \leq P_{max} \tag{11d}$$

Correspondingly, the solving of CTOP-P providing knowledge related to edge processing for the target problem. It is worth noting that the relationship between auxiliary tasks and target problem. In a single multitasking setting, the CTOP-F and the CTOP-P have similar fitness landscapes to the target problem, respectively. Meanwhile, the CTOP-F and the CTOP-P complement each other. For the same purpose but with different motivations, the auxiliary tasks can guide the solver to travel different areas of search space. The knowledge transfer can be carried out to promote the efficient search for the target problem. Most importantly, the insights offered by optimizing the local processing overhead and edge processing overhead help to make a task offloading decision. The CTOP-O can be defined based on the CTOP-F and CTOP-P as follows.

$$CTOP-O: \min_{O} \sum_{l=1}^{M_k'} [O_{l,k} Q_{l,k}^{edge} + (1-O_{l,k}) Q_{l,k}^{local}] \tag{12}$$

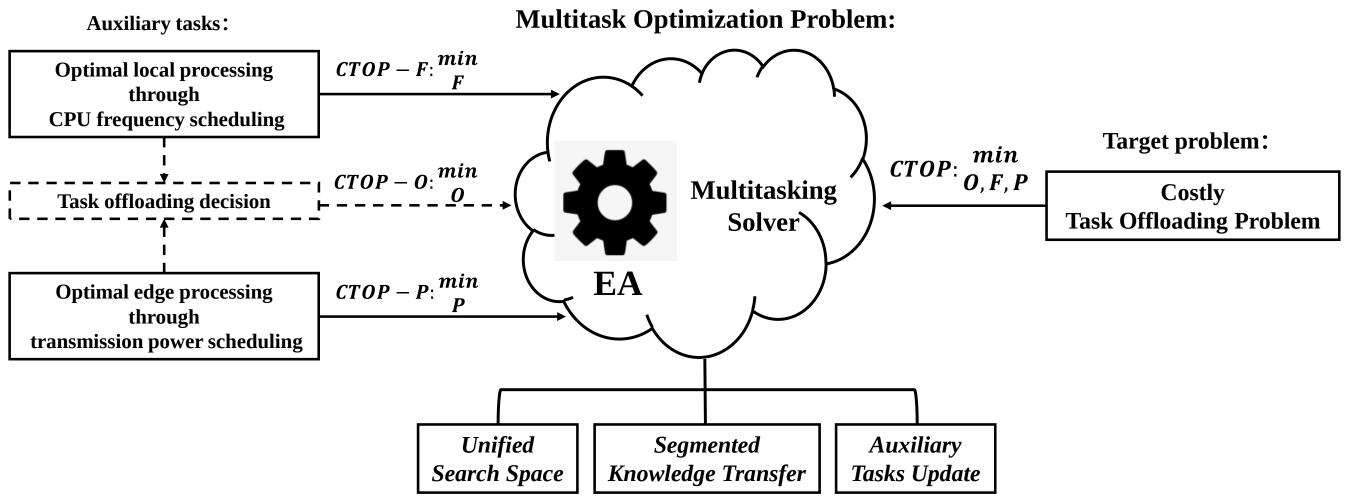
s.t.

$$C1: k \in \{1,2,\dots,N\}, M_k' \leq M_k \tag{12a}$$

$$C2: O_{i,j} \in \{0,1\} \tag{12b}$$

Making a task offloading decision needs to analyze the overheads of local processing and edge processing. In general, mobile users may prefer to choose the way with less overhead to process their computation tasks. Therefore, the searching of task offloading decision variables can be potentially obtained through CTOP-F and CTOP-P. The $Q_{l,k}^{edge}$ denotes the edge processing overhead obtain by CTOP-P and the $Q_{l,k}^{local}$ denotes the local processing overhead obtained by CTOP-F. If $Q_{l,k}^{edge} < Q_{l,k}^{local}$, then $O_{l,k} = 1$, otherwise $O_{l,k} = 0$. The task offloading decision obtained by CTOP-O is helpful for the target problem to decide whether the computation task is to be offloaded or processed locally.

As described above, we expect that the introduction of related and cheap auxiliary tasks in MTO will result in good scalability. It is critical to transfer knowledge from auxiliary tasks to the target problem to promote its efficient search. Because auxiliary tasks have fewer dimensions than the target problem, they consume less solution evaluation resource than the target problem. When the searching for auxiliary tasks can provide useful information to the target problem, i.e., the transfer of knowledge is effective, the search process can be



Multitask Optimization Problem:
Auxiliary tasks:
Optimal local processing through CPU frequency scheduling
$CTOP-F$: $\min_F$
Task offloading decision
$CTOP-O$: $\min_O$
Optimal edge processing through transmission power scheduling
$CTOP-P$: $\min_P$
Multitasking Solver
EA
Target problem:
$CTOP$: $\min_{O,F,P}$
Costly Task Offloading Problem
Unified Search Space
Segmented Knowledge Transfer
Auxiliary Tasks Update

Fig. 2. The illustration of solving CTOP with MTO.

considered cheaper comparing to directly addressing such costly target problem.

## IV. The Proposed Method

### A. Multitasking Framework for CTOP

The MTO framework of the proposed method is shown in Fig. 2. Here, the formulated CTOP is regarded as the target problem in the multitask evolution setting, the ultimate goal of a multitasking solver is to find the optimal solution for the target problem. The auxiliary tasks proposed in Section III are constructed to help improve the search of the target problem, that is reflected on using simple and related auxiliary tasks to efficiently assist costly target problems for cheaper and faster evolutionary iteration, thereby reducing computational cost and achieving convergence speedup. In particular, the evolved auxiliary tasks have complementary evolutionary trials to each other. The presence of different but complementary influencing factors may shape a favorable search guidance for the evolution of the target problem.

Noted that the component tasks in the multitask setting have different search spaces [1]. It is an effective means to encompass each of the individual search spaces for different component tasks into a unified search space. In other words, the candidate solutions corresponding to different component tasks are encoded into a vector with the same dimensionality. In such a condition, a shared pool of genetic material can be used to cover the genetic building blocks corresponding to different component tasks, thus facilitating the discovery and utilization of available genetic material in a seamless way. Furthermore, based on the consideration of commonalities and differences between the auxiliary tasks and the target problem, we divide the vectors encoded in a unified representation corresponding to the auxiliary tasks into source decision variables and extended decision variables. The source decision variables correspond to the solution of the auxiliary task, while the extended decision variables correspond to the additional part.

In a multitasking environment, the constructed auxiliary tasks influence the search trajectory of target problem mainly through knowledge transfer. For effective multitasking, it is crucial to look for useful evolutionary information. Many related approaches discussed individual-based knowledge transfer and inter-task crossover knowledge transfer. In this framework, one of our main contributions is the proposal of a novel segmented transfer strategy which facilitates the utilization of useful evolutionary information in unified search space from auxiliary tasks. If the knowledge transfer is effective, the source decision variables seem to be better suited to enhance local search and accelerate convergence, while the extended decision variables focus on maintaining diversity and improving global search.

Although the introduction of auxiliary tasks helps to reduce the difficulty of evolutionary search, they also call for additional computing resources. Different from previous studies that tended to treat each component task equally and allocate the same level of computing resources to them, we propose to generate different auxiliary tasks at different evolutionary stages. Due to the simple and low-dimension nature of the auxiliary tasks, only a small number of computing resources are needed. To use computational resources reasonably and improve the flexibility of multitasking, the allocation of resources is achieved through the update of auxiliary tasks. When an auxiliary task cannot provide more knowledge for evolutionary search of CTOP, a new auxiliary task is generated. By dynamically updating auxiliary tasks and reasonably transferring knowledge from them, the efficiency of multitasking can be significantly improved.

Considering possible transfer conflicts between auxiliary tasks, where positive and negative transfer may simultaneously exist. Rational filtering for them during the transfer process can retain useful information that facilitates evolutionary search. In addition, adaptive updating of auxiliary tasks in a multitasking environment also can effectively reduce the incidence of transfer conflicts.

### B. Multitask Evolutionary Task Offloading Algorithm

In this subsection, we intend to solve the CTOP by using the aforementioned EMT framework. We first transform the CTOP into a multi-task optimization problem by introducing the constructed auxiliary tasks, and then design an effective EMT

**Algorithm 1:** Pseudocode of METOA

```
Input:
Total population size - $NP$
Target problem CTOP – $TC$
Output:
The optimized solution of CTOP
1.  Set $t = 1$
2.  Initialize target population $C^t$ for $TC$ of size $NP/2$
3.  Evaluate candidate individuals in $C^t$
4.  Generate auxiliary tasks $TA = \{TA_j\}_{j=1}^2$ based on
    CTOP-F and CTOP-P in Eq. (10-11), respectively
5.  Initialize auxiliary populations $P^t = \{P_j^t\}_{j=1}^2$ for $TA$ of
    size $NP/4$, respectively
6.  Evaluate candidate individuals in $P^t$
7.  While stopping criteria are not satisfied do
8.    Segmented knowledge transfer via Algorithm 2
9.    Uniformly divide the $C^t$ into $\{C_j^t\}_{j=1}^2$
10.   For $j = 1$ to 2 do
11.     Perform assortative mating on $C_j^t \cup P_j^t$ to generate
        offspring $CP_j^t$
12.     Determine the imitation task of offspring in $CP_j^t$ based
        on vertical cultural transmission
13.     Evaluate the candidate individuals in $CP_j^t$
14.   End for
15.   Carry out elitist selection to form $C^{t+1}$, $P^{t+1} =$
      $\{P_j^{t+1}\}_{j=1}^2$
16.   Update auxiliary tasks via Algorithm 3
17.   Set $t = t + 1$
18. End while
```

algorithm to search for solutions. As shown in Algorithm 1, the proposed algorithm, namely multitask evolutionary task offloading algorithm (METOA), is described in detail. At the beginning of the iteration, a target population is initialized for the CTOP, in which each candidate individual corresponds to a solution of CTOP in the unified search space. Then, there are two different auxiliary tasks, which are generated according to the CTOP-F and the CTOP-P described in Section III. In METOA, each type of auxiliary tasks is offered an auxiliary population that works in parallel in the iteration process. To balance the exploration and exploitation of auxiliary populations and target population, as well as to rationalize the complementarity between auxiliary populations, each type of auxiliary task generates only one component task at a time to participate in the evolutionary in multitasking environment.

During the procedure of evolution, the candidate individuals in the auxiliary populations perform explicit knowledge transfer across tasks by Algorithm 2. The segmented knowledge transfer screen useful evolutionary information from different auxiliary tasks, with the aim of increasing the diversity of the target population and improving the fitness of candidate individuals in target population. In the offspring production cycle, the target population is uniformly divided into two subpopulations, then each of which produces offspring with an auxiliary population based on the standard evolutionary operators. The assortative mating and vertical cultural transmission [1] implicitly conduct genetic transfer across domains, where genetic material created for one component task can be copied into chromosomes associated with other component tasks. After offspring individuals are evaluated, elite candidate individuals are selected to form the next generation of the populations. Finally, considering that the auxiliary tasks are easily solved, the corresponding populations converge and diverge quickly, sometimes with stagnation. Therefore, the auxiliary tasks are updated by Algorithm 3, which dynamically generates auxiliary tasks that lead to different search biases in a multitasking environment, while rationally allocating computational resources to different auxiliary tasks.

Fig. 3 shows an example of proposed METOA with segmented knowledge transfer and auxiliary tasks update. Although the proposed algorithm is similar to the co-evolutionary algorithm in terms of the utilization of multiple populations, there are several essential differences between them. In co-evolutionary algorithm, subproblems are solved independently in subpopulations and the complete solution is obtained by combining representative individuals of each subpopulation. In contrast, the METOA is based on the framework of evolutionary multitasking, where the individuals of auxiliary tasks help the target problem to evolve search through knowledge transfer. The source and extended decision variables are available as different knowledge and the auxiliary tasks can be updated according to the search status.

## C. *Segmented knowledge transfer*

The knowledge transfer is an important issue in evolutionary multitasking and a promising way to improve the problem solving for the CTOP. Unlike previous works that focused on individual-based knowledge transfer and inter-task crossover knowledge transfer, the proposed algorithm allows for

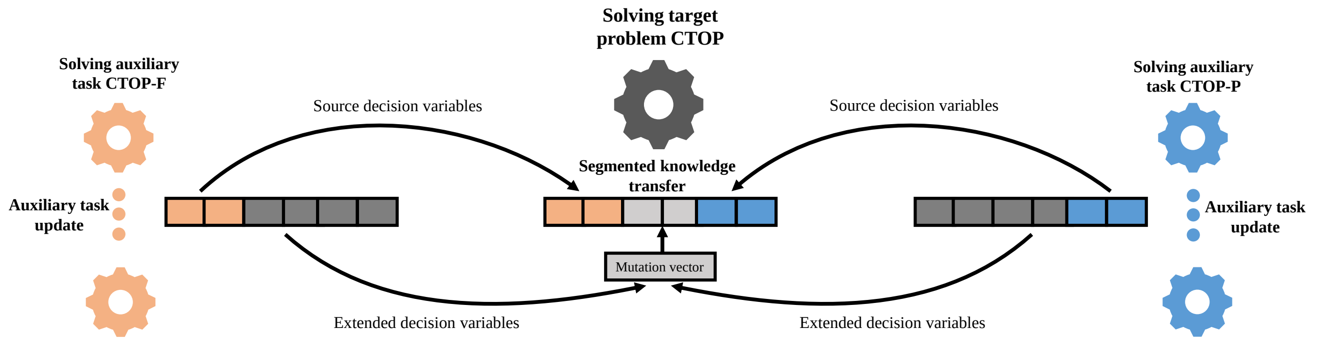

Solving target problem CTOP
Solving auxiliary task CTOP-F
Solving auxiliary task CTOP-P
Source decision variables
Source decision variables
Segmented knowledge transfer
Auxiliary task update
Auxiliary task update
Mutation vector
Extended decision variables
Extended decision variables

Fig. 3. The example of proposed METOA with segmented knowledge transfer and auxiliary tasks update.

knowledge transfer in segments. In the unified search space, the candidate individuals corresponding to the auxiliary tasks transfer knowledge to candidate individuals in target population based on source and extended decision variables.

As shown in Algorithm 2, each auxiliary population randomly provides a candidate individual for optimizing the selected candidate individual in target population. Noted that the decision variables in $p_j$ include source decision variables $p_j^s$ and extended decision variables $p_j^e$. For source decision variables of each candidate individual, they simultaneously replace the corresponding decision variables of the selected candidate individual $c$ to generate an intermediate individual $c(p_j^s)$. If positive transfer has occurred, this intermediate individual is retained in the target population. In addition, the extended decision variables need to be screened to facilitate positive transfer. For the purpose of increasing the population diversity, the extended decision variables are first extracted to generate a mutation vector $V^e$, as shown below.

$$V^e = c^e + \gamma * rand * (p_2^e - p_1^e) \tag{13}$$

where $\gamma$ is the scaling factor, $p_1^e$ and $p_2^e$ are the extended decision variables of candidate individuals $p_1$ and $p_2$, respectively. For each element in $V^e$, the ones that are verified to be beneficial will be transferred to the selected candidate individual $c$. To avoid high function evaluation cost, the Algorithm 2 performs the search for only one candidate individual in each iteration.

Since auxiliary tasks and target problem have locally similar fitness landscapes, such that the transfer of source decision variables would be likely to perform fast local search for target problem. Meanwhile, the transfer of extended decision variables from promising candidate individuals in auxiliary tasks facilitates the search of the target problem.

**Algorithm 2:** Segmented knowledge transfer

```
Input:
Target population - $C^t$
Auxiliary populations - $P^t$
Scaling factor - $\gamma$
Output:
The improved candidate individual of target population
1.  Randomly select candidate individuals $p_1, p_2$ from
    $P^t = \{P_j^t\}_{j=1}^2$, respectively
2.  Randomly select a candidate individual $c$ from $C^t$
3.  For $j = 1$ to 2 do
4.    Transfer the $p_j^s$ to the candidate individual $c$ to obtain
      $c(p_j^s)$
5.    If $c(p_j^s)$ better than $c$ then
6.      $c(p_j^s) \rightarrow c$
7.    End if
8.  End for
9.  Extract extended decision variables $p_1^e, p_2^e$ to generate
    mutation vector $V^e$ based on Eq. (13)
10. For each $V_i^e$ in $V^e$ do
11.   Transfer the $V_i^e$ to the candidate individual $c$ to
      obtain $c(V_i^e)$
12.   If $c(V_i^e)$ better than $c$ then
13.     $c(V_i^e) \rightarrow c$
14.   End if
15. End for
```

**Algorithm 3:** Auxiliary task update

```
Input:
Auxiliary populations at generation $t$ - $P^t$
Auxiliary populations at generation $t+1$ - $P^{t+1}$
Convergence coefficient – $\beta$
Output:
The updated auxiliary tasks
1. For $j = 1$ to 2 do
2.   Compute $\sigma_j = RSME(P_j^t, P_j^{t+1})$ based on Eq. (14)
3.   If $(\sigma_j \leq \beta)$ then
4.     Generate new auxiliary task $TA_j$ based on Eq. (10-11)
5.     Initialize auxiliary population $P_j^t$ for $TA_j$
6.     Evaluate candidate individuals in $P_j^t$
7.   End if
8. End for
```

### *D. Auxiliary tasks update*

The proposed algorithm allows the dynamic exploitation of auxiliary tasks to facilitate the search of the target problem. Since auxiliary tasks are easily handled, the auxiliary populations may converge or stagnate after several generations of evolution. In such a condition, it is difficult for the target population to obtain useful evolutionary information from the auxiliary populations, while unnecessary evaluation resources may be wasted on auxiliary tasks that have already diverged. Therefore, the generation of different auxiliary tasks in the evolutionary process makes full use of limited evaluation resources to facilitate cross-domain search. As shown in Algorithm 3, the degree of change in auxiliary populations is used to determine whether they need to be updated. During the evolutionary iteration, auxiliary populations are constantly changing and eventually stabilize. Based on the fitness of population members, the change in the auxiliary population can be observed according to the following equation.

$$RMSE(P_j^t, P_j^{t+1}) = \sqrt{\frac{1}{NP/4}\sum_{i=1}^{NP/4}(fit(p_i)_j^t - fit(p_i)_j^{t+1})^2} \tag{14}$$

The change between $P_j^t$ and $P_j^{t+1}$ is measured by the individual fitness $fit(p_i)_j^t$ and $fit(p_i)_j^{t+1}$. If the $\sigma_j$ is less than a given convergence coefficient $\beta$, then new auxiliary tasks will be generated and given evaluation resources to participate in evolution. The purpose of auxiliary task update is to provide diverse knowledge of local optimization to guide the evolutionary search of the target problem. The new auxiliary task is generated by updating the parameters $k$ and $M_k^{'}$ in CTOP-F or CTOP. Each update of the auxiliary task indicates that the population of candidate individuals will shift to search for promising solutions for $M_k^{'}$ mobile devices in region $k$, thus achieving quick local optimization.

## V. Experiment Studies

### *A. Experimental Setup*

The experiment studies were conducted based on the system model presented in Section III. We consider a simulated MEC network for task offloading, where $N$ SBSs are randomly distributed in a 500 m $*$ 500 m area. There is an MBS located in the center of the MEC network, which is equipped with a MEC server. In the default settings, the number of SBSs is set to 10 and the computation frequency of the MEC server is 4 GHz. There are 100 mobile devices uniformly scattered over the MEC network, each mobile device has a computation task to be processed. The data size and the number of required CPU cycles of the computation task are randomly generated in the range of [400, 1200] KB and $[0.1 * 10^9, 1 * 10^9]$ cycles, respectively. The maximum CPU frequency and the maximum transmission power of mobile devices are 1 GHz and 23 dBm. respectively. The noise power is set to -100 dBm and the uplink bandwidth is 2 MHz. Unless otherwise indicated, when discussing the effect of a parameter on the performance of the system model, the value of this parameter is variable while all other parameters are set as default values.

To demonstrate the effectiveness of the proposed algorithm for solving the CTOP, we compare the METOA with several state-of-the-art multitask algorithms as well as the single-task algorithms. The well-known multitask algorithms, that are, MFEA [1] and MFEA-II [2], are considered as the baseline algorithms, for using auxiliary tasks to solve CTOP. To verify the benefits of introducing auxiliary task for optimization, we further implemented two other variants of MFEA, namely are MFEA-F and MFEA-P, which share the same replication as MFEA, but with different settings of auxiliary task. Specifically, MFEA-F and MFEA-P only involve auxiliary tasks CTOP-F and CTOP-P, respectively. For the same reason, METOA-F, METOA-P, MFEA-II-F and MFEA-II-P are also considered for discussion. In addition, four representative single-task algorithms, including HGPCA [35], PSO [36], DE [37] and GA [38], are involved in the comparison with METOA. The default settings of the compared algorithms are given as follows.

(1) Population size: $NP = 100$.

(2) Maximum function evaluations: $maxFEs = 4000$.

(3) Independent number of runs: $Runs = 20$.

(4) The parameters for MFEA: Crossover probability $p_c = 1$, mutation probability $p_m = 1/D$, and random mating probability $rmp = 0.3$.

(5) The parameters for MFEA-II: Crossover probability $p_c = 1$, mutation probability $p_m = 1/D$, and random mating probability based on online $rmp$ learning.

(6) The parameters for the proposed METOA: Scaling factor $\gamma = 0.5$, convergence coefficient $\beta = 0.1$, crossover probability $p_c = 1$, and mutation probability $p_m = 1/D$.

(7) The parameters for the HGPCA: Crossover probability $p_c = 1$, mutation probability $p_m = 1/D$, Inertia weight $w_I = 0.4$, acceleration instants $c_1 = c_2 = 1.5$.

(8) The parameters for the PSO: Inertia weight $\mathrm{w_I} = 0.6$, learn factors $c_1 = c_2 = 2$.

(9) The parameters for the DE: Mutation factor $F = 0.9$, crossover probability $c_R = 0.8$.

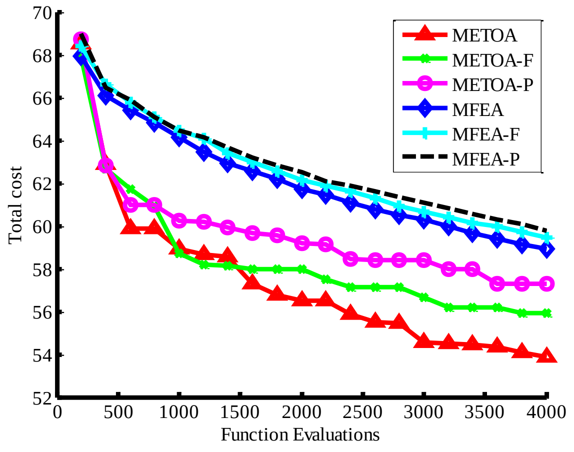
(10) The parameters for the GA: Crossover probability $p_c = 1$, mutation probability $p_m = 1/D$.

Note that in the numerical analysis, the Wilcoxon rank sum test at a significance level of 0.05 is performed to evaluate the contrast results. The symbols "+" or "-" indicates that the corresponding algorithm has a better or worse average performance than METOA, respectively. The symbol "≈" denotes that the corresponding algorithm and METOA have a similar average performance.

## *B. Multitask Algorithms with Auxiliary Tasks*

Since the introduction of auxiliary tasks can bring efficient multitasking for handling complex optimization problems, we analyze and compare the search performances of the MFEA, MFEA-II and the proposed METOA based on different auxiliary tasks in optimizing CTOP.

As shown in Fig. 4, an overview of the convergence performance of each algorithm during the evolutionary process is provided. There are obvious differences in the total cost of task offloading with different algorithms. It can be observed from Fig. 4 that the convergence performance of METOA outperforms all other compared algorithms. Meanwhile, the METOA, MFEA-II and MFEA are able to achieve better convergence performance compared to their variants, respectively. This result, as expected, demonstrates the knowledge transfer from auxiliary tasks in MTO does contribute to convergence acceleration and solution finding. When compared against the MFEA-F and the MFEA-P, the MFEA obtains better results. This is mainly due to the fact that MFEA employs both auxiliary tasks CTOP-F and CTOP-P,



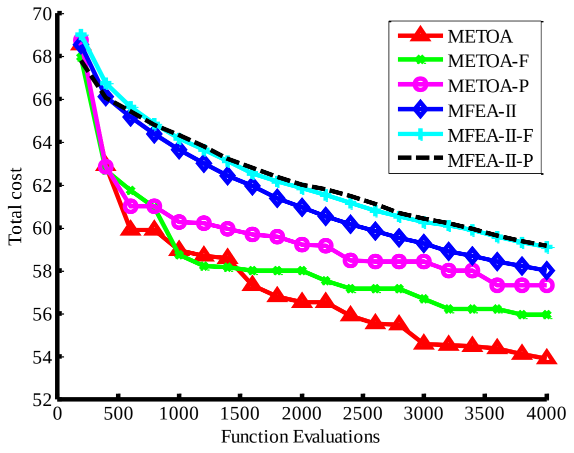


Fig. 4. The convergence trends of METOA, METOA-F, METOA-P, MFEA, MFEA-F, MFEA-P, MFEA-II, MFEA-II-F and MFEA-II-P.

which possess the advantages of strong local search. Similarly, the MFEA-II has been observed to outperform the MFEA-II-F and the MFEA-II-P. As mentioned before, the only difference between the algorithms and their corresponding variants is the configuration of the auxiliary tasks, the obtained results again confirmed the effectiveness of introducing auxiliary tasks in solving CTOP.

It was noticed that METOA and its variants (METOA-F and METOA-P) obtained better results than MFEA and their variants (MFEA-F and MFEA-P), respectively. Moreover, the MFEA-II and its variants (MFEA-II-F and MFEA-II-P) also obtained better results than MFEA and their variants (MFEA-F and MFEA-P), respectively. The potential reason is the gap in the efficiency of the genetic transfer. Although cross-task genetic transfer occurred during the evolution of MFEA, its vulnerability to negative transfer makes them difficult to enhance the search. The MFEA-II improves the process of genetic transfer based on online RMP learning. In addition, METOA, METOA-F and METOA-P are good at obtaining positive transfer for search. Specifically, METOA and its variants (METOA-F and METOA-P) can quickly optimize the total cost of the CTOP in the early stages of evolution. On the one hand, cheap auxiliary tasks are optimized rapidly due to their simple nature. On the other hand, the proposed segmented knowledge transfer enhances the local search capability using the source decision variables learned by the optimized auxiliary task, while improving the global search capability using the extended decision variables. In the subsequent evolutionary search, the auxiliary tasks are updated adaptively. With beneficial cross-domain transfer of high-quality genetic material, it tends to guide the cost function of CTOP converge smoothly around the global optimum.

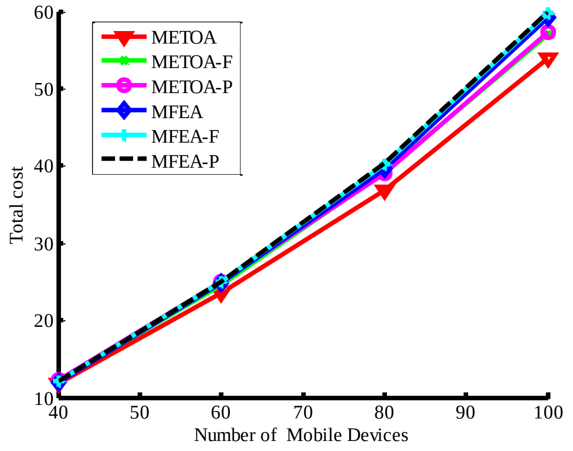


Fig. 5. The total cost of task offloading under different number of mobile devices.

Table II shows the mean value and standard deviation of the total cost on CTOP instances of all the multitask comparison algorithms according to 20 independent runs, where the best result is highlighted.

The proposed METOA has been observed to outperform all other algorithms for different number of function evaluations on almost all CTOP instances. In particular, the proposed METOA achieves better solution quality when the number of mobile devices increases, compared to MFEA and MFEA-II, on CTOP instances with different numbers of mobile devices. Moreover, it also can be observed that, the proposed METOA can maintain excellent performance when the computational resources (function evaluations) are constrained. For example, for an CTOP instance with 40 mobile devices, the proposed METOA with 2000 function evaluations achieves competitive result against MFEA with 4000 function evaluations. As shown in Fig. 5, we plot the total cost versus the number of mobile devices. This further validates that the proposed METOA is capable of adaptively optimizing task offloading as the number of mobile devices rise.

In general, it can be observed from the obtained results above that introducing cheaper auxiliary tasks and exploiting knowledge from there to speed up the problem solving of CTOP not only improves the search of multitasking, but also enhances the scalability of the solver to cope with large-scale problems.

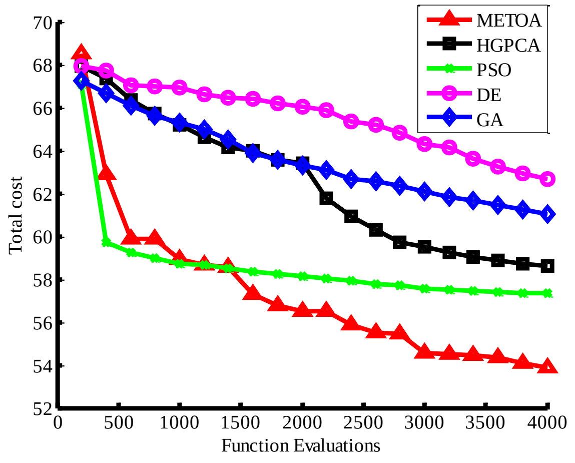


Fig. 6. The convergence trends of METOA, HGPCA, PSO, DE and GA.

### C. Comparison with Single-Task Algorithms

For a comprehensive performance evaluation, we also discuss the performance of the proposed METOA contrast to the single-tasking algorithms in different CTOP instances.

Fig. 6 shows the curves of the total cost on CTOP instance based on 20 independent runs of METOA, HGPCA, PSO, DE, and GA with 4000 function evaluations. The results show that the proposed METOA has better convergence performance than HGPCA, PSO, DE, and GA during the procedure of evolution. We noted that although the PSO converges faster than METOA, HGPCA, DE, and GA in in the early stages of evolution, it soon plunges into a local optimum. This indicates that METOA can effectively overcome the issue of local optimum, and achieves a good balance between convergence and diversity. The reason may be that the generated auxiliary tasks contribute to obtain a good diversity of the target problem. When the search is trapped in a local optimum, the update of the auxiliary tasks can drive the exploitation of other areas of the search space. Table III shows the mean value, best value and standard deviation of total cost of METOA, HGPCA, PSO, DE, and GA on CTOP instances with different numbers of mobile devices and SBSs, in which the best results are marked in bold. In the multiple independent runs, it can be seen that the best solution found by the proposed algorithm is better than other algorithms, which demonstrates the advantage of the proposed algorithm to find high-quality solutions. The results show that the transfer of source decision variables can accelerate the convergence of the algorithm due to the high similarity between the auxiliary task and the target problem. Meanwhile, the transfer of the extended decision variables can improve the outcomes of the algorithm that terminates prematurely due to local optimum. Moreover, it can be

Table II

THE AVERAGE VALUE AND STANDARD DEVIATION OF TOTAL COST OF TASK OFFLOADING OBTAINED BY METOA, METOA-F, METOA-P, MFEA, MFEA-F and MFEA-P ON CTOP INSTANCES, WHERE THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

| Algorithms | Function Evaluations | Number of Mobile Devices | | | |
|---|---|---|---|---|---|
| | | 40 | 60 | 80 | 100 |
| METOA | 2000 | **1.2094e+01** (2.3807e-01) | **2.4848e+01** (7.0865e-01) | **3.7769e+01** (1.1458e+00) | **5.5161e+01** (9.2405e-01) |
| | 4000 | **1.1758e+01** (1.7024e-01) | **2.3478e+01** (3.1101e-01) | **3.6830e+01** (5.9951e-01) | **5.3982e+01** (1.5661e+00) |
| | 6000 | **1.1583e+01** (1.3299e-01) | **2.3262e+01** (3.5270e-01) | **3.5975e+01** (7.0538e-01) | **5.2318e+01** (1.0289e+00) |
| METOA-F | 2000 | 1.2632e+01 - (2.7213e-01) | 2.5291e+01 ≈ (4.3374e-01) | 3.9421e+01 - (1.1589e+00) | 5.7958e+01 - (1.2971e+00) |
| | 4000 | 1.2085e+01 - (1.3483e-01) | 2.4615e+01 - (4.0855e-01) | 3.9164e+01 - (1.1208e+00) | 5.6966e+01 - (1.6780e+00) |
| | 6000 | 1.2088e+01 - (1.3612e-01) | 2.4252e+01 - (2.8128e-01) | 3.7417e+01 - (7.4720e-01) | 5.5114e+01 - (1.5006e+00) |
| METOA-P | 2000 | 1.2543e+01 - (2.3688e-01) | 2.5679e+01 - (7.8266e-01) | 4.0386e+01 - (9.7280e-01) | 5.8896e+01 - (1.4340e+00) |
| | 4000 | 1.2250e+01 - (2.0732e-01) | 2.4932e+01 - (5.7432e-01) | 3.9039e+01 - (8.7909e-01) | 5.7244e+01 - (1.6063e+00) |
| | 6000 | 1.1996e+01 - (1.7837e-01) | 2.4337e+01 - (4.7890e-01) | 3.7790e+01 - (8.6645e-01) | 5.5362e+01 - (1.4026e+00) |
| MFEA | 2000 | 1.2442e+01 - (1.0352e-01) | 2.5997e+01 - (1.9544e-01) | 4.1817e+01 - (5.3147e-01) | 6.2012e+01 - (9.5408e-01) |
| | 4000 | 1.2014e+01 - (9.7236e-02) | 2.4826e+01 - (3.0732e-01) | 3.9639e+01 - (6.8410e-01) | 5.9165e+01 - (7.8502e-01) |
| | 6000 | 1.1779e+01 - (1.1172e-01) | 2.4015e+01 - (2.6456e-01) | 3.8123e+01 - (3.7728e-01) | 5.6969e+01 - (8.9715e-01) |
| MFEA-F | 2000 | 1.2568e+01 - (7.5329e-02) | 2.6085e+01 - (3.9408e-01) | 4.2190e+01 - (5.3363e-01) | 6.2059e+01 - (1.0260e+00) |
| | 4000 | 1.2128e+01 - (1.1772e-01) | 2.5017e+01 - (2.5685e-01) | 4.0124e+01 - (6.2806e-01) | 5.9716e+01 - (7.0933e-01) |
| | 6000 | 1.1943e+01 - (1.2160e-01) | 2.4329e+01 - (3.1334e-01) | 3.8809e+01 - (4.5461e-01) | 5.7632e+01 - (5.9805e-01) |
| MFEA-P | 2000 | 1.2539e+01 - (2.1739e-01) | 2.6207e+01 - (3.9268e-01) | 4.2120e+01 - (8.2937e-01) | 6.2591e+01 - (5.1515e-01) |
| | 4000 | 1.2128e+01 - (1.2327e-01) | 2.5015e+01 - (1.6712e-01) | 4.0271e+01 - (4.8329e-01) | 5.9860e+01 - (8.2737e-01) |
| | 6000 | 1.1883e+01 - (1.0490e-01) | 2.4433e+01 - (2.4128e-01) | 3.8689e+01 - (3.5247e-01) | 5.7996e+01 - (8.6785e-01) |
| MFEA-II | 2000 | 1.2329e+01 - (1.2629e-01) | 2.5627e+01 - (2.4021e-01) | 4.1332e+01 - (7.4524e-01) | 6.0956e+01 - (6.6838e-01) |
| | 4000 | 1.1857e+01 ≈ (1.1299e-01) | 2.4570e+01 - (2.6410e-01) | 3.8896e+01 - (5.3974e-01) | 5.7982e+01 - (9.9825e-01) |
| | 6000 | 1.1636e+01 ≈ (5.7249e-02) | 2.3674e+01 - (2.9410e-01) | 3.7797e+01 - (2.7983e-01) | 5.6218e+01 - (7.1525e-01) |
| MFEA-II-F | 2000 | 1.2504e+01 - (1.4705e-01) | 2.6151e+01 - (2.3974e-01) | 4.1752e+01 - (4.7920e-01) | 6.2054e+01 - (5.6797e-01) |
| | 4000 | 1.2061e+01 - (1.4016e-01) | 2.4908e+01 - (2.4502e-01) | 3.9836e+01 - (3.9772e-01) | 5.9085e+01 - (6.9965e-01) |
| | 6000 | 1.1761e+01 - (1.2547e-01) | 2.4074e+01 - (2.3342e-01) | 3.8425e+01 - (4.3768e-01) | 5.7138e+01 - (5.0945e-01) |
| MFEA-II-P | 2000 | 1.2524e+01 - (4.7692e-02) | 2.6249e+01 - (2.7793e-01) | 4.1481e+01 - (4.0246e-01) | 6.2283e+01 - (6.5650e-01) |
| | 4000 | 1.2100e+01 - (9.5371e-02) | 2.4968e+01 - (1.7748e-01) | 3.9877e+01 - (5.9537e-01) | 5.9119e+01 - (5.2419e-01) |
| | 6000 | 1.1803e+01 - (1.1301e-01) | 2.4150e+01 - (1.4334e-01) | 3.8709e+01 - (3.6382e-01) | 5.6970e+01 - (3.9182e-01) |

observed that the proposed METOA obtains a better average performance on most CTOP instances. Expectedly, the obtained average total cost of all the algorithms increases as more mobile devices are served. Meanwhile, with an increasing number of SBSs, the higher average total cost can be caused by all the algorithms because of the increased interferences the mobile device suffered. When the number of mobile devices and SBSs is small, we can see that the PSO achieves better results than METOA, HGPCA, DE and GA. Although the METOA achieves slightly lower results than PSO, it outperforms HGPCA, DE and GA on most of the CTOP instances. Nevertheless, we can see from Fig. 7 that, the

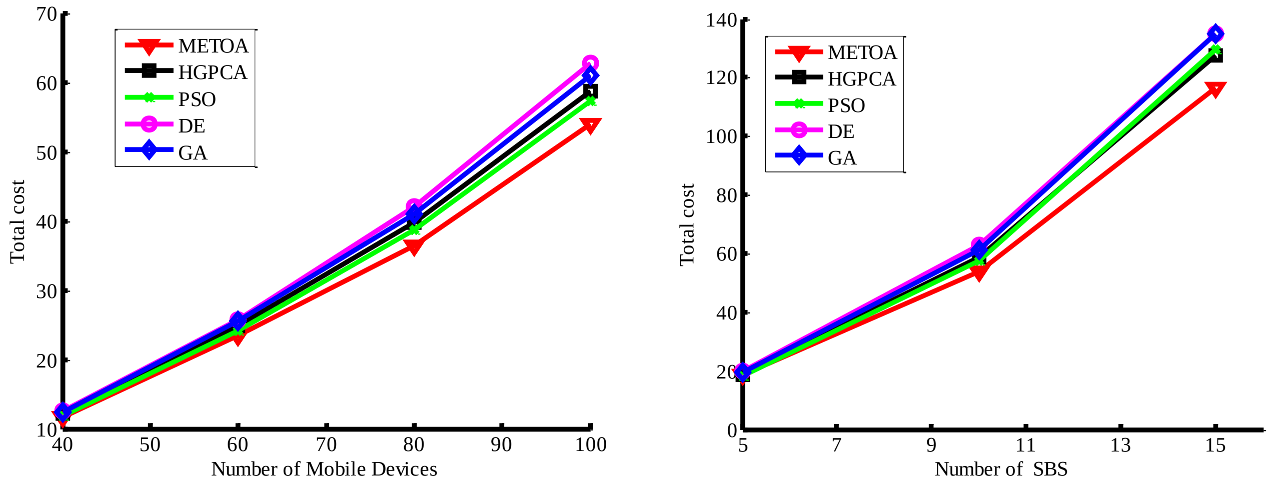


Fig. 7. The total cost of task offloading under different number of mobile devices and SBSs, respectively.

Table III

THE AVERAGE VALUE, BEST VALUE AND STANDARD DEVIATION OF TOTAL COST OF TASK OFFLOADING OBTAINED BY METOA, HGPCA, PSO, DE and GA ON CTOP INSTANCES, WHERE THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

| **Number of Mobile Devices** | **Number of SBS** | **Algorithms** | | | | |
|---|---|---|---|---|---|---|
| | | **METOA** | **HGPCA** | **PSO** | **DE** | **GA** |
| 40 | 5 | 5.2213e+00<br>5.1988e+00<br>(1.7816e-02) | 5.2603e+00 –<br>5.2300e+00<br>(1.8750e-02) | **5.2024e+00 +**<br>**5.1758e+00**<br>(1.9942e-02) | 5.3713e+00 –<br>5.3396e+00<br>(2.4659e-02) | 5.3081e+00 –<br>5.2786e+00<br>(1.7786e-02) |
| | 10 | **1.1702e+01**<br>**1.1446e+01**<br>(1.4386e-01) | 1.2205e+01 –<br>1.2007e+01<br>(1.2952e-01) | 1.1883e+01 ≈<br>1.1556e+01<br>(2.4559e-01) | 1.2617e+01 –<br>1.2292e+01<br>(1.4717e-01) | 1.2420e+01 –<br>1.2271e+01<br>(9.4962e-02) |
| | 15 | **3.1767e+01**<br>**3.0907e+01**<br>(6.2586e-01) | 3.6878e+01 –<br>3.6318e+01<br>(5.7907e-01) | 3.9056e+01 –<br>3.6384e+01<br>(2.0996e+00) | 3.4393e+01 –<br>3.2594e+01<br>(8.3541e-01) | 3.8338e+01 –<br>3.7646e+01<br>(4.0270e-01) |
| 60 | 5 | 9.2638e+00<br>9.1974e+00<br>(4.3654e-02) | 9.3892e+00 –<br>9.3068e+00<br>(5.8511e-02) | **9.1871e+00 +**<br>**9.1599e+00**<br>(2.0298e-02) | 9.7216e+00 –<br>9.6085e+00<br>(7.8282e-02) | 9.5520e+00 –<br>9.4976e+00<br>(3.6647e-02) |
| | 10 | **2.3477e+01**<br>**2.2808e+01**<br>(3.0134e-01) | 2.4859e+01 –<br>2.4457e+01<br>(2.2177e-01) | 2.4184e+01 –<br>2.3440e+01<br>(5.3173e-01) | 2.5770e+01 –<br>2.5525e+01<br>(3.5373e-01) | 2.5523e+01 –<br>2.5170e+01<br>(3.2043e-01) |
| | 15 | **5.8940e+01**<br>**5.6148e+01**<br>(2.3378e+00) | 6.8816e+01 –<br>6.7015e+01<br>(1.2775e+00) | 6.9599e+01 –<br>6.5379e+01<br>(3.7367e+00) | 6.7427e+01 –<br>6.5148e+01<br>(1.5385e+00) | 7.1087e+01 –<br>7.0329e+01<br>(5.4559e-01) |
| 80 | 5 | 1.3546e+01<br>1.3416e+01<br>(8.4269e-02) | 1.3638e+01 ≈<br>1.3491e+01<br>(1.5114e-01) | **1.3356e+01 +**<br>**1.3323e+01**<br>(3.9581e-02) | 1.4374e+01<br>1.4105e+01<br>(1.9545e-01) | 1.4163e+01<br>1.3987e+01<br>(1.1151e-01) |
| | 10 | **3.6421e+01**<br>**3.5770e+01**<br>(6.4641e-01) | 3.9666e+01 –<br>3.8696e+01<br>(5.9005e-01) | 3.8743e+01 –<br>3.7578e+01<br>(9.6275e-01) | 4.1962e+01 –<br>4.0317e+01<br>(1.0101e+00) | 4.0968e+01 –<br>4.0207e+01<br>(6.6808e-01) |
| | 15 | **8.4774e+01**<br>**8.2270e+01**<br>(2.2755e+00) | 9.9090e+01 –<br>9.7385e+01<br>(9.6888e-01) | 9.6154e+01 –<br>9.1758e+01<br>(4.4446e+00) | 1.0154e+02 –<br>9.7978e+01<br>(2.5617e+00) | 1.0262e+02 –<br>1.0172e+02<br>(8.0160e-01) |
| 100 | 5 | 1.8569e+01<br>1.8346e+01<br>(1.3873e-01) | 1.8546e+01 ≈<br>1.8428e+01<br>(9.2339e-02) | **1.8235e+01 +**<br>**1.8188e+01**<br>(2.7692e-02) | 1.9978e+01 –<br>1.9736e+01<br>(2.1405e-01) | 1.9621e+01 –<br>1.9287e+01<br>(1.9804e-01) |
| | 10 | **5.3883e+01**<br>**5.1626e+01**<br>(2.3127e+00) | 5.8598e+01 –<br>5.7108e+01<br>(7.7233e-01) | 5.7341e+01 –<br>5.5258e+01<br>(1.6363e+00) | 6.2645e+01 –<br>6.0466e+01<br>(1.3440e+00) | 6.1002e+01 –<br>5.9763e+01<br>(6.5879e-01) |
| | 15 | **1.1651e+02**<br>**1.1033e+02**<br>(4.1832e+00) | 1.2742e+02 –<br>1.2402e+02<br>(1.8425e+00) | 1.2975e+02 –<br>1.1803e+02<br>(9.2492e+00) | 1.3496e+02 –<br>1.3331e+02<br>(1.3718e+00) | 1.3483e+02 –<br>1.3331e+02<br>(9.2928e-01) |

performance of the PSO drops significantly with an increasing number of mobile devices and SBSs, as does HGPCA, DE and GA. In contrast, the METOA manages to achieve much better performance than all the single-task algorithms in such a condition, which demonstrates the good scalability of METOA.

## D. *Effectiveness Analysis of Knowledge Transfer*

To verify the effectiveness of knowledge transfer in METOA, we further analyzed the synergistic effects of the proposed segmented knowledge transfer and auxiliary tasks update in evolutionary search.

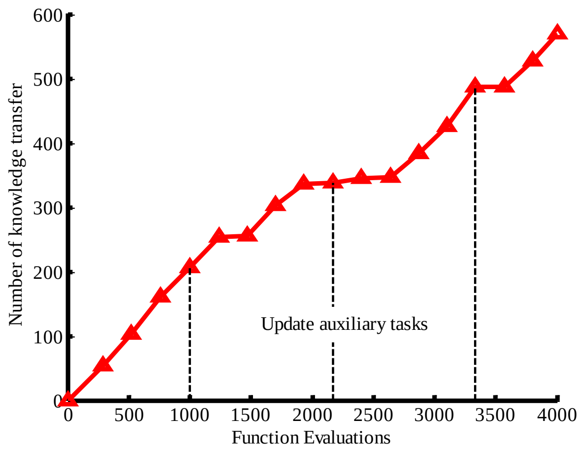


Fig. 8. The number of knowledge transfer from the auxiliary tasks to the CTOP.

According to the proposed strategies in Section IV, we first recorded the number of knowledge transfer from the auxiliary tasks to the CTOP. Then, we investigated the updates of the auxiliary tasks in different evolutionary stages. Notably, the knowledge transfer mentioned here indicates a positive transfer that the transfer of genetic material can make an improvement for the obtained solution of CTOP. Fig. 8 shows the rising trend of the number of knowledge transfer throughout the evolutionary process. It can be observed that the knowledge transfer from auxiliary tasks provides a great help to CTOP in the early stages of evolution, which is consistent to the results obtained in Fig. 4 and Fig. 6. This is probably because the cheap auxiliary tasks are easier to be handle than CTOP, which can provide useful evolutionary information quickly for CTOP, resulting in a rapid discovery of high-quality solutions. However, this is clearly not the case in the middle stages of evolution. As the auxiliary tasks gradually converge and eventually stagnate, genetic transfer from the auxiliary tasks is hardly effective. To tackle this, auxiliary tasks are updated to change the direction of evolution and improve the diversity of population. Therefore, in Fig. 8, the number of knowledge transfer has been observed to rise again, after the new auxiliary tasks are generated. There is a similar situation in the later stages of evolution. As a result, the METOA successfully exploits the positive transfer from the auxiliary tasks, while circumventing the distress caused by evolutionary stagnation.

Further, the knowledge transfer from the auxiliary tasks to the CTOP mainly includes source decision variables transfer and extended decision variables transfer. Within a single transfer cycle, the transferred source and extended decision variables are from the same chromosome, the difference being that the former uses direct transfer and the latter uses indirect transfer. To observe the effect of knowledge transfer more clearly, in Fig. 9, we plot the success rate (ratio of the number of positive transfers to the total number of transfer) of transferring the source and extended decision variables from the auxiliary tasks to the CTOP throughout the evolutionary process. We can see that the transfer of both the decision variables is valid for the CTOP. As expected, in the early stages of evolution, for knowledge transfer, the source decision variables have a higher efficiency than the extended decision variables. This is due to the high similarity between the auxiliary tasks and CTOP, the transfer of source decision variables is able to accelerate convergence process. Complementary to source decision variables, the expanded decision variables are transferred to maintains a good diversity. In summary, the knowledge transfer in METOA is constructive for the evolutionary search in the CTOP.

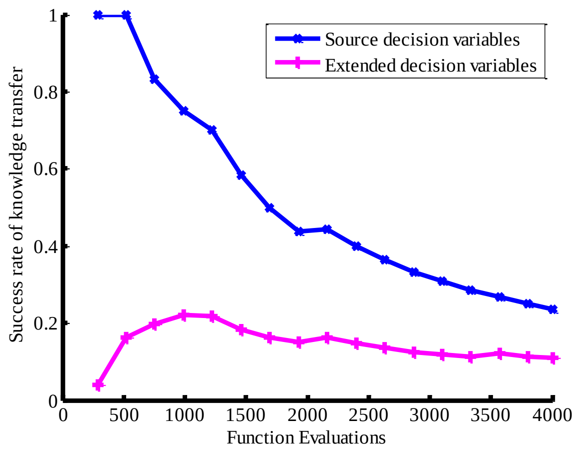


Fig. 9. The success rate of transferring the source and extended decision variables from the auxiliary tasks to the CTOP, respectively.

## VI. Conclusion

In this article, we handled the costly task offloading problem by introducing related and cheap auxiliary tasks into the proposed evolutionary multitasking framework. With the help of knowledge extraction and transfer from auxiliary tasks, the target problem can be quickly optimized. Moreover, we proposed a novel multitask evolutionary task offloading algorithm for solving the costly task offloading problem by using segmented knowledge transfer and auxiliary task update. Experimental results show that the proposed algorithm is capable of good performance in convergence, diversity and scalability, which demonstrates that the knowledge transfer from constructed auxiliary tasks can indeed effectively facilitate the search performance of the costly task offloading problem.

In future work, we plan to study how to construct available auxiliary tasks for complex optimization problems. Meanwhile, when there are multiple auxiliary tasks, how the auxiliary tasks should work in collaboration is an issue to be investigated.

## Declaration of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] T. X. Tran, A. Hajisami, P. Pandey and D. Pompili, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," IEEE Communications Magazine, vol. 55, no. 4, pp. 54-61, April 2017.

[2] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7457-7469, Aug. 2020.

[3] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[4] Gupta, Abhishek, and Y. S. Ong. "Back to the Roots: Multi-X Evolutionary Computation," Cognitive Computation, vol. 11, no. 1, pp. 1–17, Jan. 2019.

[5] F. Song, H. Xing, S. Luo, D. Zhan, P. Dai and R. Qu, "A Multiobjective Computation Offloading Algorithm for Mobile-Edge Computing," IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8780-8799, Sept. 2020.

[6] L. Pan, X. Liu, Z. Jia, J. Xu and X. Li, "A Multi-objective Clustering Evolutionary Algorithm for Multi-workflow Computation Offloading in Mobile Edge Computing," IEEE Transactions on Cloud Computing, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[7] J. Zhang et al., "Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2633-2645, Aug. 2018.

[8] A. Younis, T. X. Tran and D. Pompili, "Energy-Latency-Aware Task Offloading and Approximate Computing at the Mobile Edge," 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2019, pp. 299-307.

[9] A. Gupta, Y. Ong and L. Feng, "Multifactorial Evolution: Toward Evolutionary Multitasking," IEEE Transactions on Evolutionary Computation, vol. 20, no. 3, pp. 343-357, June 2016.

[10] K. K. Bali, Y. -S. Ong, A. Gupta and P. S. Tan, "Multifactorial Evolutionary Algorithm With Online Transfer Parameter Estimation: MFEA-II," IEEE Transactions on Evolutionary Computation, vol. 24, no. 1, pp. 69-83, Feb. 2020.

[11] A. Gupta, L. Zhou, Y. S. Ong, Z. Chen and Y. Hou, "Half a Dozen Real-World Applications of Evolutionary Multitasking, and More," IEEE Computational Intelligence Magazine, vol. 17, no. 2, pp. 49-66, May 2022.

[12] N. Zhang, A. Gupta, Z. Chen and Y. -S. Ong, "Evolutionary Machine Learning With Minions: A Case Study in Feature Selection," IEEE Transactions on Evolutionary Computation, vol. 26, no. 1, pp. 130-144, Feb. 2022.

[13] X. Ma et al., "Enhanced Multifactorial Evolutionary Algorithm With Meme Helper-Tasks," IEEE Transactions on Cybernetics, vol. 52, no. 8, pp. 7837-7851, Aug. 2022.

[14] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, and C. Yue, "An evolutionary multitasking optimization framework for constrained multi-objective optimization problems," IEEE Transactions on Evolutionary Computation, vol. 26, no. 2, pp. 1–15, April. 2022.

[15] P. Dai, K. Liu, X. Wu, Y. Liao, V. C. S. Lee and S. H. Son, "Bandwidth Efficiency and Service Adaptiveness Oriented Data Dissemination in Heterogeneous Vehicular Networks," IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp. 6585-6598, July 2018.

[16] X. Wang, Z. Ning and L. Wang, "Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4568-4578, Oct. 2018.

[17] Z. Ning, J. Huang and X. Wang, "Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities," IEEE Wireless Communications, vol. 26, no. 1, pp. 87-93, February 2019.

[18] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4005-4018, June 2019.

[19] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268-4282, Oct. 2016.

[20] T. Q. Dinh, J. Tang, Q. D. La and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," IEEE Transactions on Communications, vol. 65, no. 8, pp. 3571-3584, Aug. 2017.

[21] J. Zhang et al., "Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2633-2645, Aug. 2018.

[22] C. Wang, F. R. Yu, C. Liang, Q. Chen and L. Tang, "Joint Computation Offloading and Interference Management in Wireless Cellular Networks with Mobile Edge Computing," IEEE Transactions on Vehicular Technology, vol. 66, no. 8, pp. 7432-7445, Aug. 2017.

[23] L. Feng et al., "An empirical study of multifactorial PSO and multifactorial DE," 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 921-928.

[24] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong and Tan Puay Siew, "Linearized domain adaptation in evolutionary multitasking," 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1295-1302.

[25] Z. Liang, W. Liang, Z. Wang, X. Ma, L. Liu and Z. Zhu, "Multiobjective Evolutionary Multitasking With Two-Stage Adaptive Knowledge Transfer Based on Population Distribution," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 52, no. 7, pp. 4457-4469, July 2022.

[26] Z. Liang, H. Dong, C. Liu, W. Liang and Z. Zhu, "Evolutionary Multitasking for Multiobjective Optimization With Subspace Alignment and Adaptive Differential Evolution," IEEE Transactions on Cybernetics, vol. 52, no. 4, pp. 2096-2109, April 2022.

[27] M. Gong, Z. Tang, H. Li and J. Zhang, "Evolutionary Multitasking With Dynamic Resource Allocating Strategy," in IEEE Transactions on Evolutionary Computation, vol. 23, no. 5, pp. 858-869, Oct. 2019.

[28] H. Xu, A. K. Qin and S. Xia, "Evolutionary Multitask Optimization With Adaptive Knowledge Transfer," in IEEE Transactions on Evolutionary Computation, vol. 26, no. 2, pp. 290-303, April 2022.

[29] J. Ding, C. Yang, Y. Jin and T. Chai, "Generalized Multitasking for Evolutionary Optimization of Expensive Problems," IEEE Transactions on Evolutionary Computation, vol. 23, no. 1, pp. 44-58, Feb. 2019.

[30] F. Zhang, Y. Mei, S. Nguyen, M. Zhang and K. C. Tan, "Surrogate-Assisted Evolutionary Multitask Genetic Programming for Dynamic Flexible Job Shop Scheduling," IEEE Transactions on Evolutionary Computation, vol. 25, no. 4, pp. 651-665, Aug. 2021.

[31] X. Hou et al., "Reliable Computation Offloading for Edge-Computing-Enabled Software-Defined IoV," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7097-7111, Aug. 2020.

[32] N. Zhang, A. Gupta, Z. Chen and Y. -S. Ong, "Evolutionary Machine Learning With Minions: A Case Study in Feature Selection," IEEE Transactions on Evolutionary Computation, vol. 26, no. 1, pp. 130-144, Feb. 2022.

[33] K. Qiao et al., "Dynamic Auxiliary Task-Based Evolutionary Multitasking for Constrained Multi-objective Optimization," in IEEE Transactions on Evolutionary Computation, 2022.

[34] Z. Chen, A. Gupta, L. Zhou and Y. -S. Ong, "Scaling Multiobjective Evolution to Large Data With Minions: A Bayes-Informed Multitask Approach," in IEEE Transactions on Cybernetics, 2022.

[35] 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," in IEEE/ACM Transactions on Networking, vol. 26, no. 6, pp. 2651-2664, Dec. 2018.

[36] Y. Wen, Q. Zhang, H. Yuan and J. Bi, "Multi-Stage PSO-Based Cost Minimization for Computation Offloading in Vehicular Edge Networks," 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), 2021, pp. 1-6.

[37] X. Li, G. Zhang, X. Zheng and S. Hua, "Delay Optimization Based on Improved Differential Evolutionary Algorithm for Task Offloading in Fog Computing Networks," 2020 International Conference on Wireless Communications and Signal Processing (WCSP), 2020, pp. 109-114.

[38] D. Ye, M. Wu, S. Tang and R. Yu, "Scalable Fog Computing with Service Offloading in Bus Networks," 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), 2016, pp. 247-251.