Multifactorial Memetic Algorithm with Adaptive Auxiliary Tasks for Service Migration Optimization in 5G Mobile Edge Computing

Guo Li, Zhaobo Liu, Ling Liu, Zexuan Zhu, *Senior Member, IEEE*

***Abstract*—In high-speed fifth-generation (5G) cellular networks, Mobile Edge Computing (MEC) is tasked with assigning mobile users to the right servers to minimize the response time. A crucial aspect of MEC is optimizing the migration of services based on user mobility, which is a complex problem posing significant challenges to conventional optimization methods. To tackle this problem, we develop a multifactorial memetic algorithm with adaptive auxiliary tasks or MFMA-AAT for short. MFMA-AAT solves the target service migration optimization problem and an adaptively selected auxiliary task simultaneously, where the auxiliary task is a simplified version of the target problem to guide the search towards promising regions faster via knowledge transfer. Multiple auxiliary tasks are pre-constructed based on the distribution of the mobile users and the one with best improvement at each generation is selected for knowledge transfer. A community detection based memetic operator is also introduced to accelerate the local convergence of the proposed algorithm. Experimental results demonstrate that MFMA-AAT is more efficient than traditional service migration approaches and other state-of-the-art multifactorial evolutionary algorithms.**

***Keywords*-Multifactorial evolutionary algorithm; Memetic algorithm; Community detection; Service migration; Mobile edge computing**

I. Introduction

With the exponential growth of connected end devices and the constant emergence of 5G applications, delivering a user-centric Quality of Service (QoS) is becoming increasingly crucial [1]. The long physical transmission distance can impede the reduction of round-trip delay [2], making traditional cloud computing insufficient for providing low-latency QoS. To address this issue, Mobile Edge Computing (MEC) steps in, bringing computing resources closer to the mobile user equipment (UE) and reducing response time [3]. MEC is not just another form of cloud computing on distributed edge servers, like cloudlet and fog computing [4], [5]. It is QoS-aware, dynamically allocating and adjusting resources including CPU, storage, and bandwidth to prevent QoS degradation for the UE [6]. With the server being located close to the base station and UE, 5G MEC networks offer ultra-low response times [7].

Intuitively, it seems like keeping the UE service profile close to the user by placing it on the nearest MEC server and following the user's movements would lead to the lowest latency. However, the actual user experience of QoS is impacted by communication, migration, and calculation latencies. Current profile tracking algorithms aim to minimize communication latency [8], but this optimization comes with a trade-off in terms of migration latency. In addition, migration can lead to hotspot congestion on MEC servers, further increasing calculation latency [9]. This makes service migration of mobile computing capabilities a crucial area of research [10]. Service migration needs to tackle the challenge of selecting the right MEC server among a pool of candidates to place the UE profile for an ongoing service [11].

Traditional service migration algorithms are mainly based on Game Theory (GT) [12] that tends to get trapped in local optima. Recently, evolutionary algorithms (EAs) have been employed in the optimization of service migration thanks to their good global search capability [13]. Nevertheless, traditional EAs solve one single service migration optimization problem at a time from scratch, which might suffer from slow convergence as the scale and complexity of the target problem increase. Evolutionary multitasking [14]–[17] can serve as a candidate solution to this issue by solving the target service migration optimization task with additional auxiliary task(s) simultaneously. An auxiliary task usually is a related task or a simplified version of the original target problem. Solving the auxiliary task can provide helpful knowledge to assist the solve of the target problem in evolutionary multitasking via knowledge transfer. For example, multifactorial evolutionary algorithms (MFEAs) [14], [18] solving multiple related optimization tasks simultaneously has been demonstrated to achieve better performance than their single-task counterparts. MFEA-MVD [19] employs multiobjectivization via decomposition to generate effective auxiliary tasks to improve the solving of the primary task. Shang et al. [20] solved vehicle routing problems by introducing simplified auxiliary tasks in evolutionary multitasking, showing that knowledge transfer between tasks can improve the evolutionary search. Yang et al. [21] solved operational indices optimization in beneficiation processes with multitasking multiobjective evolutionary algorithm by constructing less-accurate models as auxiliary tasks to enhance the convergence of the most accurate model.

The effectiveness of the constructed auxiliary task(s) is pivotal to the success of evolutionary multitasking. Most of the existing evolutionary multitasking algorithms use constant auxiliary task throughout the evolutionary search, which might be less efficient. Using different auxiliary tasks in different stages of the search could be more beneficial to solve the

G. Li, Z. Liu, L. Liu and Z. Zhu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail:szuliguo@szu.edu.cn; liuzhaobo@szu.edu.cn; liulingcs@szu.edu.cn; zhuzx@szu.edu.cn).

target problem. To this end, this work introduces MFMA-AAT, a multifactorial memetic algorithm with adapts auxiliary tasks to solve service migration optimization problem in 5G MEC. Using K-means clustering, MFMA-AAT initially groups UEs based on their locations in MEC and sets auxiliary tasks corresponding to these clusters to optimize resource allocation in each group. In each iteration, MFMA-AAT dynamically assigns higher selection probability to the auxiliary task that contributes most to the convergence of the main task. MFMA-AAT also imposes memetic operator based on community detection and local search to accelerate the local convergence. Particularly, using cosine similarity in community detection [22], MFMA-AAT pre-selects a few MEC servers as a neighborhood and then uses hill climbing to find the optimal solution.

MFMA-AAT boosts the efficiency of optimization with dynamically selected auxiliary tasks that enhance the search with adaptive knowledge. The convergence of the primary task is accelerated and the search space is compacted via the memetic operator. Empirical evaluations show that MFMA-AAT outperforms the traditional GT methods and other state-of-the-art MFEAs in identifying the optimal service placement solution. The contributions of this study are summarized as follows:

- An adaptive selection mechanism of auxiliary task is explored in evolutionary multitasking, where K-means clustering is introduced to constructed auxiliary tasks that are then selected based on their contributions.
- A memetic operator is designed to accelerate the convergence of the algorithm in local regions by leveraging community detection and local search.
- The resultant algorithm MFMA-AAT is extensively investigated in different scales of service migration optimization problems to reveal the strength and weakness.

The remainder of this paper is organized as follows: An overview of the background and related work on the subject is provided in Section II. Section III introduces the details of MFMA-AAT. The experimental results are described in Section IV. Section V summarizes the main conclusions of this study.

II. Background

This section provides the preliminaries of the service migration optimization problem in 5G MEC and an overview on the related work of EA-based MEC and MFEAs. For the conveninece of the reader, the notations adopted in this paper are summarized in Table I.

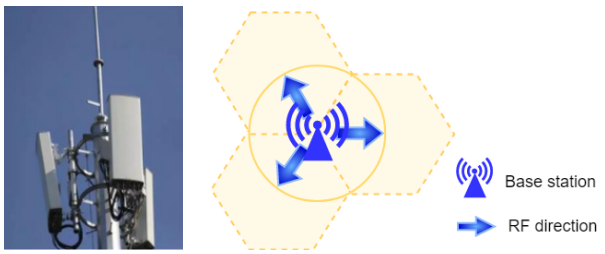
A. Problem Formulation

In 5G cellular network, a typical base station has three radio frequency (RF) antennas to broadcast signals, as illustrated in Fig. 1. A group of base stations employs a hexagonal cellular network structure to maximize coverage area while minimizing frequency collisions, as depicted in Fig. 2.

As a UE moves between base stations, the UE's service profile can also be migrated between MEC servers. We define a simulated MEC network that contains η UEs,

TABLE I
MAIN NOTATIONS USED IN THIS PAPER

Symbol	Description
η	Number of UE
ω	Number of MEC servers
U	UE set $\{\mathrm{U}_1, \mathrm{U}_2, ..., \mathrm{U}_i, ..., \mathrm{U}_\eta\}$
P	UE profile set $\{\mathrm{P}_1, \mathrm{P}_2, ..., \mathrm{P}_i, ..., \mathrm{P}_\eta\}$
B	Base station set $\{\mathrm{B}_1, \mathrm{B}_2, ..., \mathrm{B}_j, ..., \mathrm{B}_\omega\}$
M	MEC server set $\{\mathrm{M}_1, \mathrm{M}_2, ..., \mathrm{M}_j, ..., \mathrm{M}_\omega\}$
κ	Number of K-means groups
G	K-means group set $\{\mathrm{G}_1, \mathrm{G}_2, ..., \mathrm{G}_k, ..., \mathrm{G}_\kappa\}$
T	Time slot set $\{..., t\text{-}1, t, t\text{+}1, ...\}$
$\mathrm{U}_i^j(t)$	Indicator of UE i being connected (value 1) or not (value 0) to base station j
$\mathrm{P}_i^j(t)$	Indicator of user profile U_i placing (value 1) or not (value 0) at MEC server j
G_k	0-1 array mask $[\theta_1, \theta_2, ..., \theta_i, ..., \theta_\eta]$ where θ_i is an indicator of UE i belonging (value 1) or not (value 0) K-means group k
$\mathrm{U}_i^{\mathrm{G}_k}$	Indicator of UE i belonging (value 1) or not (value 0) to K-means group k
ρ_j	Computing capacity of MEC server j
$\mu_j(t)$	Number of UE in MEC server j
$R_i^j(t)$	Computation requirement of UE i in MEC server j
$Q_i^j(t)$	Queued number of required calculations by UE i in MEC server j
$L_i^\alpha(t)$	Communication latency of UE i
$L_i^\beta(t)$	Migration latency of UE i
$L_i^\gamma(t)$	Calculation latency of UE i
$L_i(t)$	User-perceived latency of UE i
$\mathbf{x}$	A placement solution for all profiles, also a chromosome
Ω	The solution space
$\mathbb{L}(\mathbf{x})$	The average user-perceived latency in the MEC system
λ	Number of chromosomes in a population
P_c	Probability of crossover
P_m	Probability of mutation
P_l	Probability of local search
$\mathbf{X}$	A population, also a solution set $\{\mathbf{x_1}, \mathbf{x}_2, ..., \mathbf{x}_v, ..., \mathbf{x}_\lambda\}$
c_v^i	The i-th gene in the chromosome $\mathbf{x}_v$, also indicates $\mathrm{P}_i^j(t)$
$F(\mathbf{x})$	Main task objective function, $F(\mathbf{x})$=$\mathbb{L}(\mathbf{x})$
$A_k(\mathbf{x})$	Auxiliary task objective function, $A_k(\mathbf{x})$=$\mathbb{L}(\mathbf{x}.*G_k)$
ϵ	Select probability array on κ auxiliary tasks $[\epsilon_1, \epsilon_2, ..., \epsilon_k, ..., \epsilon_\kappa]$, $\sum_1^\kappa \epsilon_k$=1
Max^G	Maximum number of generations
Max^E	Maximum number of evaluations
Max^T	Maximum running time
$\mathbf{X}_n$	The population in the n-th iteration, $n \in [1, \mathrm{Max}^G]$
$\delta_k(n)$	Drop rate of auxiliary task k in the n-th iteration
$\mathbb{L}(\mathbf{X}_n^\star)$	The user-perceived latency of the optimal chromosome in population $\mathbf{X}_n$
ξ_{ij}	A practical attribute vector of P_i placing at M_j
$\xi_{ij}^\star$	An ideal attribute vector of P_i placing at M_j
Φ_i	A MEC server community of U_i



(a) (b)

Fig. 1. Structure of base station: (a) Photograph of the wireless base station including three radio frequency (RF) devices. (b) Three RF devices can cover 360 degrees of wireless communication.

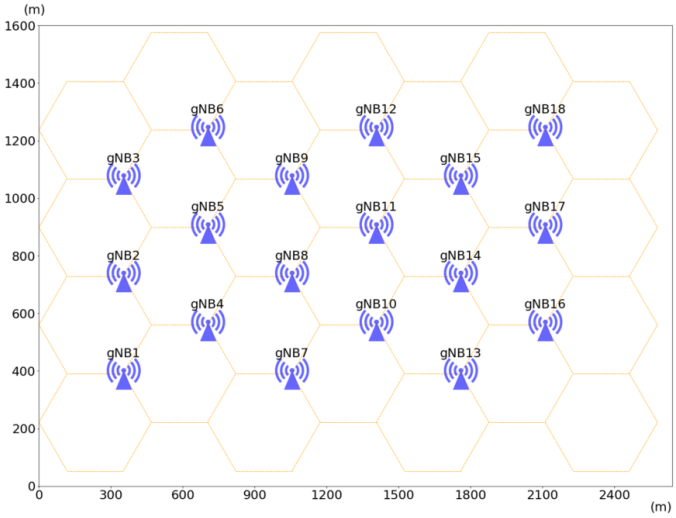


Fig. 2. Cellular network. The base stations are generally arranged in a hexagonal cell layout. In our simulation platform, each base station is configured with an MEC server.

TABLE II
HOP TIMES BETWEEN MEC SERVERS

	M_1	M_2	M_3	...	M_{16}	M_{17}	M_{18}
M_1	0	1	2	...	4	4	5
M_2	1	0	1	...	4	4	4
M_3	2	1	0	...	4	4	4
M_4	3	2	1	...	4	4	4
M_5	1	1	2	...	3	3	4
M_6	2	1	1	...	3	3	3
M_7	3	2	1	...	3	3	3
M_8	2	2	3	...	2	3	4
M_9	2	2	2	...	2	2	3
M_{10}	3	2	2	...	2	2	2
M_{11}	4	3	2	...	3	2	2
M_{12}	3	3	3	...	1	2	3
M_{13}	3	3	3	...	1	1	2
M_{14}	4	3	3	...	2	1	1
M_{15}	4	4	4	...	1	2	3
M_{16}	4	4	4	...	0	1	2
M_{17}	4	4	4	...	1	0	1
M_{18}	5	4	4	...	2	1	0

i.e., U=$\{U_1, U_2, \ldots, U_i, \ldots, U_\eta\}$, and ω base stations, i.e., B=$\{B_1, B_2, \ldots, B_j, \ldots, B_\omega\}$. Each UE has a one-to-one correspondence to user profile, thus the indices of user profile set P=$\{P_1, P_2, \ldots, P_i, \ldots, P_\eta\}$ are the same as that of U. For the sake of simplicity, each base station is paired with an MEC server. Similarly, the indices of MEC server set M=$\{M_1, M_2, \ldots, M_j, \ldots, M_\omega\}$ are consistent with B. To model UE mobility, a divided time slot set T=$\{..., t\text{-}1, t, t\text{+}1, ...\}$ is given for indicating continuous periods. Each UE profile is restricted to only one MEC server during a time slot. In each time slot, the UE's location is recorded only once, and the UE's service profile remains unchanged on an MEC server. Thus, the constraint for the UE location $U_i^j(t)$ is defined as follows:

$$\begin{aligned} \sum_{j=1}^{\omega} U_i^j(t) = 1, \quad \forall t, \\ U_i^j(t) \in \{0, 1\}, \end{aligned} \tag{1}$$

where $U_i^j(t)$ indicates whether U_i is under base station B_j in time slot t. Specifically, $U_i^j(t) = 1$ if U_i is under B_j in time slot t, and $U_i^j(t) = 0$ otherwise. Similarly, the relationships for the service profiles are as follows:

$$\begin{aligned} \sum_{j=1}^{\omega} P_i^j(t) = 1, \quad \forall t, \\ P_i^j(t) \in \{0, 1\}, \end{aligned} \tag{2}$$

where $P_i^j(t) = 1$ denotes P_i exists at MEC server M_j in time slot t, and $P_i^j(t) = 0$ otherwise.

In MEC systems, the user-perceived latency is a crucial metric for QoS. The user-perceived latency is a result of the interplay between three latency factors: communication (α), migration (β), and calculation (γ).

The communication latency L^α is comprised of two components. One is the radio propagation latency from the base station to the UE, which is considered a constant, σ. The other component is the transmission latency between the MEC server and the base station. The transmission latency is determined by the number of hops from the profile-hosting MEC server to the server belonging to the UE's associated base station. If the profile-hosting MEC server belongs to the UE's associated base station, the transmission latency is zero. According to [8], we assume that the UE profile transmission latency between neighboring base stations is equal to one hop. Radio propagation latency, σ, is equal to half the hop time. In our experiments, the hop time is taken as the basic unit of the user-perceived latency. In the MEC network shown in Fig. 2, the corresponding hop times between the MEC servers in the network are listed in Table II.

The communication latency L^α of UE i is defined as

$$L_i^\alpha(t) = \mathrm{Hop}(\,\mathrm{IdxU}(U_i, t),\ \mathrm{IdxP}(P_i, t)\,) + \sigma, \quad \forall t, \tag{3}$$

where the functions $\mathrm{IdxU}(U_i, t)$ and $\mathrm{IdxP}(P_i, t)$ indicate the index of located base station and served MEC server of UE i in time slot t, respectively, and the function $\mathrm{Hop}(\,\mathrm{IdxU}(U_i, t),\ \mathrm{IdxP}(P_i, t)\,)$ denotes the transmission latency of two locations. As the indices of the base station and MEC are the same, the hop time in Table II indicates the transmission latency.

The migration latency occurs when a UE profile need to migrate to a new MEC server. Migration latency is proportional to the migration distance, namely, the number of hops between two MEC servers. Thus, the migration latency L^β of UE i is defined as follows:

$$L_i^\beta(t) = \mathrm{Hop}(\,\mathrm{IdxP}(P_i, t\text{-}1),\ \mathrm{IdxP}(P_i, t)\,), \quad \forall t, \tag{4}$$

where $\mathrm{IdxP}(P_i, t)$ and $\mathrm{IdxP}(P_i, t\text{-}1)$ denote the indices of current and previous MEC servers, respectively. The function $\mathrm{Hop}(\,\mathrm{IdxP}(P_i, t\text{-}1),\ \mathrm{IdxP}(P_i, t)\,)$ indicates the profile migration time, which can also be retrieved from Table II.

The calculation latency L^{γ} of a UE's requirements on an MEC server is based on the amount of computation needed and the resources allocated to the UE by the MEC server. We define the computation requirement of UE i at time slot t as $R_i^j(t)$ and the computing capacity of server j as ρ_j, which is considered constant due to its hardware configuration. The number of UEs on server j at time slot t is represented by $\mu_j(t)$. Under an average allocation strategy, each user's computing resources on server j is calculated as $\rho_j/\mu_j(t)$. The calculation for a UE includes current requirements and any remaining computation from previous time slots, with the condition that queued calculations cannot be negative. If the computation resources exceed the requirements, the remaining queued calculations are set to 0. Thus, the calculation latency of UE i is defined as follows:

$$\begin{aligned} L_i^{\gamma}(t) &= \frac{R_i^j(t) + Q_i^j(t)}{\rho_j/\mu_j(t)}, \quad \forall t, \\ Q_i^j(t) &= \max(Q_i^j(t-1) + R_i^j(t) - \rho_j/\mu_j(t),\ 0), \end{aligned} \tag{5}$$

where $Q_i^j(t)$ indicates the remaining queued calculations of UE i at time slot t.

The user-perceived latency of UE i is given by:

$$L_i(t) = L_i^{\alpha}(t) + L_i^{\beta}(t) + L_i^{\gamma}(t), \tag{6}$$

where $L_i(t)$ is user-perceived latency of UE i in time slot t, $L_i^{\alpha}(t)$ is communication latency, $L_i^{\beta}(t)$ is migration latency, and $L_i^{\gamma}(t)$ is calculation latency.

We aim to optimize the average value of user-perceived latency for all UEs in an MEC system as follows:

$$\min \mathbb{L} = \frac{1}{\eta} \sum_{i=1}^{\eta} L_i(t), \quad \forall t. \tag{7}$$

B. Related Work

1) EA-Based MEC: EAs have been widely used in MEC scheduling. For example, Zhu et al. [23] proposed an edge-computing resource allocation strategy based on an improved genetic algorithm. The paper adopted integer coding, knowledge-based crossover, and mutation of population segmentation to improve the optimization ability of the genetic algorithm. Bi et al. [24] formulated a genetically simulated annealing-based particle swarm optimization to achieve joint optimization of computation offloading between a cloud data center and edge server. Ahmed et al. [25] considered parallel and sequential task offloading for multiple MEC servers. Based on the genetic algorithm, simulation results demonstrate that the sequential solution provides a dropped failure probability, and the parallel solution yields a lower latency. Liu et al. [26] presented a joint optimization objective of limited resources, higher operating costs, and certain failure probabilities in MEC, and subsequently introduced an optimized task allocation approach with a biogeography-based optimization algorithm, which is an evolutionary algorithm that can maximize the effectiveness of task allocation and reduce communication delay simultaneously. Wang et al. [27] designed a distributed clustering strategy to classify vehicles into multiple cooperative edge servers and presented an online heuristic algorithm for offloading tasks with shorter delays and maximizing system service revenue. Addad et al. [28] introduced network slicing for mobility events caused by moving end-users in a 5G network. In a network slice, the computing resources need to be re-scoped and re-provisioned, and services must be migrated to reduce system overhead and ensure low communication latency by following end-user mobility patterns. Tang et al. [29] proposed a novel offloading algorithm based on a genetic algorithm for finding the optimal offloading decision in MEC, which not only optimizes the tasks in the current network slice but also reduces the task pressure in the next network slice, which can better exploit the computing capacities of the MEC servers. Xiao et al. [30] realized MEC grouping for task offloading in MEC cooperation, and simulation results demonstrated that fast convergence can be achieved under various system parameters, such as time slots and the number of MEC servers. Although valuable results have been achieved in the optimization of MEC performance using EAs, service migration remains an immature and highly unproven technology.

2) MFEAs: MFEAs, first proposed in [14], [18], have been shown to outperform single-task EAs thanks to the introduction of knowledge transfer. The efficiency of knowledge transfer is critical to the performance of MFEAs. For example, Wei et al. [31] concluded that traditional EAs might suffer from a high computational burden and poor generalization ability in complex optimization problems, and MFEA is a candidate solution to overcoming the problem by efficient knowledge extraction across distinct optimization task domains. Zhou et al. [32] recognized that the crossover is the basis for achieving knowledge transfer in MFEA, and different crossover operators affect efficiency of knowledge transfer. Thus, they proposed MFEA-AKT to update crossover dynamically for adaptive knowledge transfer. Xu et al. [33] introduced an MFEA to solve multiple optimization tasks and avoid negative knowledge transfer between tasks. They proposed a series of adaptive knowledge transfer operators to make the best use of knowledge transfer. Liang et al. [34] considered that knowledge transfer is easily trapped in a local optimum when multitasks are highly similar and suffers from negative transfer when multitasks have low similarity. Hence, they proposed an adaptive knowledge transfer method based on the population distribution. In the first stage of knowledge transfer, an adaptive weight is used to reduce the impact of negative transfers. In the second stage of knowledge transfer, the search range of each individual is further adjusted dynamically, which can improve the population diversity and be beneficial for jumping out of local optimum. Wu et al. [35] considered the similarities among network reconstruction tasks at different component layers, developed an MFEA to improve reconstruction performance, and selectively performed knowledge transfer according to the sparsity of multilayer networks. Aritz et al. [36] proposed an adaptive MFEA used in multitask reinforcement learning. They noticed that in previous studies, tasks in reinforcement learning were mostly irrelevant. Thus, they focused on improving the knowledge transfer among related tasks and achieved high success rates on multiple tasks simultaneously. Feng et al. [37] found that

most evolutionary multitasking (EMT) algorithms exchange knowledge implicitly by crossover, so they presented an EMT algorithm based on explicit knowledge transfer among multi-tasks. The experimental results on vehicle routing benchmarks showed the efficacy of explicit knowledge transfer based on the solution.

A few MFEAs speed up the convergence by introducing auxiliary tasks with positive knowledge transfer. For instance, Ma et al. [19] proposed MFEA with multiobjectivization via decomposition to construct positive related auxiliary tasks. Its auxiliary tasks are constructed through subtask decomposition, which required certain prior knowledge and appropriate combination of subtasks. Li et al. [38] used an adaptive information transfer mechanism to solve a competitive multitasking optimization problem in which all task objectives are comparable. Shang et al. [20] used a memetic search with evolutionary multitasking to solve the vehicle routing problem, which was simultaneously performed on multiple simplified vehicle routing problems as auxiliary tasks and the original VRP as the main task, and each task will perform a local search. Wang et al. [39] employed surrogate-assisted MFEA to address the challenges of a high-dimensional search space and high computation cost optimization. A surrogate model in the joint space of the decision and scenario spaces was constructed to replace part of the expensive function evaluations. Liu et al. [40] incorporated surrogate-assisted evolutionary algorithms for optimization problems with a costly fitness evaluation. Via parallel random grouping for the costly problem, the assisted subtasks with constraint boundary can speed up the convergence.

Most of the existing MEFAs use constant auxiliary task during all stages of the search, which might lead to inefficient knowledge transfer. Selecting auxiliary tasks adaptively according to the staged search performance of the algorithm could be more favorable. In this regard, we propose the MFMA-AAT as follows.

III. Proposed MFMA-AAT

In this section, we present the details of MFMA-AAT, i.e., a novel approach combining MFEA, adaptive auxiliary tasks (AAT), and memetic operator to tackle service migration in 5G MEC systems. Fig. 3 illustrates the MFMA-AAT flowchart. MFMA-AAT as described in Alg. 1 starts by randomly generating an initial population, each chromosome's cluster groups being identified based on the UE location in the MEC network. After setting the main task as classical MFEAs, the AAT method adaptively selects auxiliary task in each generation. MFMA-AAT defines a mask string to symbolize each group, with the same number of auxiliary tasks as there are clustered groups. The multifactorial evolution stage involves crossover, mutation, computation of factorial cost, scalar fitness evaluation for the population, and selection of offspring. The evolution process is then followed by memetic operator, which performs a local search using community detection to determine the optimal individuals and update the population. The algorithm continues until either the maximum running time, fitness evaluation, or number of generations is reached.

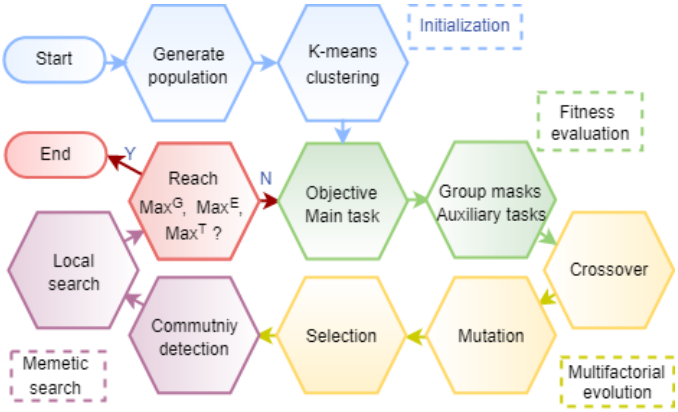


Fig. 3. Flowchart of MFMA-AAT algorithm. (i) Initialization: generate population and group UEs with K-means. (ii) Evaluation: set the main task's objective, get masks of groups according to the UE index, and set auxiliary tasks corresponding to the masks. (iii) multifactorial evolution: crossover and mutate chromosomes, compute factorial cost and get factorial rank, select elite chromosomes. (iv) Memetic search: perform community detection to generate a neighboring space, and conduct a local search to provide an enhanced population. (v) Terminate condition: the maximum Generation Max^G, the maximum fitness evaluation Max^E, the maximum running time Max^T.

A. Initialization and Evaluation Functions

To initialize a solution for service migration in MEC, we start by generating a population randomly. Each individual in the population is a chromosome encompassing a group of genes. In this context, a gene is a decision variable, where the migration decision is selecting a node from a server list for each service profile in each time slot. This constitutes a combinatorial optimization problem, and to address it, we use gene coding with an η-dimensional decision variable vector whose domain is the set of MEC server indices. Ultimately, a chromosome represents a complete profile placement solution for all UEs in the MEC system.

The initial population is a group of solutions usually represented by a matrix $\mathbf{X}$ $[\eta * \lambda]$, where λ indicates the number of chromosomes and η denotes the length of a chromosome representing the number of UE. When generating an initial population, each gene value is randomly selected from $[1, \omega]$, where ω denotes the number of MEC servers. The number of chromosomes should be even for the subsequent evolutionary operators.

The optimization objective of the main task is user-perceived latency, and the fitness evaluation function is given by

$$\min : F(\mathbf{x}) = \mathbb{L}(\mathbf{x}), \qquad s.t. \quad \mathbf{x} \in \Omega, \tag{8}$$

where $\mathbf{x}$ is a decision vector of a migration solution, Ω is the solution space, and $\mathbb{L}(\mathbf{x})$ is the average user-perceived latency in the MEC system as defined in Eq. (7).

For a randomly generated population, the initial skill factor is evenly distributed between the tasks. The tasks comprise the main task and all the auxiliary tasks. In a chromosome, a gene is from 1 to ω, which represents the profile of the UE belonging to the MEC server. To share chromosomes between the main and auxiliary tasks, we add a new gene, 0, which indicates that the UE is not in the MEC network. In the

Algorithm 1 MFMA-AAT

Input: Population size η, Number of K-means groups κ, Probability of crossover P_c, Probability of mutation P_m, Objective function $\mathbb{L}(\mathbf{x})$, Terminate conditions: $\mathrm{Max}^G, \mathrm{Max}^E, \mathrm{Max}^T$

Output: The optimal solution $\mathbf{x}^\star$

Initialize population $\mathbf{X}$: $\{\mathbf{x_1}, \mathbf{x}_2, ..., \mathbf{x}_v, ..., \mathbf{x}_\eta\}$
Initialize main task $\mathbb{L}(\mathbf{x})$
for v = 1 to η **do**
  Initialize skill factor from [0, κ]
  Compute K-means groups $\{\mathrm{G}_1, \mathrm{G}_2, ..., \mathrm{G}_k, ..., \mathrm{G}_\kappa\}$
  Initialize auxiliary tasks $A_k(\mathbf{x}) = \mathbb{L}(\mathbf{x}.*G_k)$
  Initialize auxiliary selection probability $\{\epsilon_k | \epsilon_k = 1/\kappa, k \in (1, \kappa)\}$
end for
while NOT reach G_{max} or E_{max} or T_{max} **do**
  $\mathbf{X}^{offspring} \leftarrow$ uniform_crossover($\mathbf{X}$, P_c)
  $\mathbf{X}^{offspring} \leftarrow$ swap_mutate($\mathbf{X}^{offspring}$, P_m)
  Select auxiliary task k=roulette($\{\epsilon_k | k \in (1, \kappa)\}$)
  Set auxiliary function $A_k(\mathbf{x}) = \mathbb{L}(\mathbf{x} * G_k)$
  $\mathbf{X}^{union} \leftarrow \mathbf{X} \cup \mathbf{X}^{offspring}$
  for $v = 1$ to $2 * \eta$ **do**
    if $\mathbf{x}_v.skill\_factor$==0 **then**
      $\mathbf{x}_v.factorial\_cost$ = $\mathbb{L}(\mathbf{x}_v)$
    else
      $\mathbf{x}_v.factorial\_cost$ = $A_k(\mathbf{x}_v)$
    end if
  end for
  compute_factorial_rank($\mathbf{X}^{union}$)
  compute_scalar_fitness($\mathbf{X}^{union}$)
  $\mathbf{X} \leftarrow$ select_elitist($\mathbf{X}^{union}$)
  compute_drop_rate (δ_k)
  update_selection_probability($\{\epsilon_k | k \in (1, \kappa)\}$)
  $\mathbf{X} \leftarrow$ memetic($\mathbf{X}$) (See Alg. 2)
end while
$\mathbf{x}^\star$ = select_optimal($\mathbf{X}$)

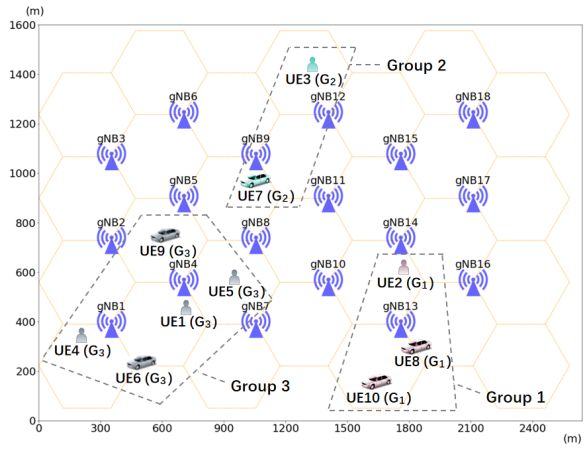


Fig. 4. The clustered groups of 10 UE. UE $[2, 8, 10]$ are in group 1, UE $[3, 7]$ are in group 2, and UE $[1, 4, 5, 6, 9]$ are in group 3.

auxiliary task, the evolutionary operation is performed only on the nonzero UE. If the K-means method divides UEs into κ groups, we set κ masks for the groups. For example, a chromosome $[4, 13, 12, 1, 7, 1, 9, 13, 4, 13]$ contains 10 UEs as depicted in Fig. 4. If the number of K-means groups is 3, the UE groups are:

$$[U_2^{13}, U_8^{13}, U_{10}^{13}],$$
$$[U_3^{12}, U_7^{9}],$$
$$[U_1^{4}, U_4^{1}, U_5^{7}, U_6^{1}, U_9^{4}],$$

where the MEC server M_9 and M_{12} are closer, M_1, M_4 and M_7 are also not far apart. The three corresponding group masks are:

$$G_1\text{: } [0, 1, 0, 0, 0, 0, 0, 1, 0, 1],$$
$$G_2\text{: } [0, 0, 1, 0, 0, 0, 1, 0, 0, 0],$$
$$G_3\text{: } [1, 0, 0, 1, 1, 1, 0, 0, 1, 0].$$

If we add the three mask vectors vertically, we get an all-ones vector $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$. The auxiliary functions are initialized according to the group of masks,

$$A_k(\mathbf{x}) = \mathbb{L}(\mathbf{x}.*G_k), \qquad s.t. \quad \mathbf{x} \in \Omega, k \in [1, \kappa] \tag{9}$$

where A_k is the auxiliary task fitness function, G_k is the mask of group k, κ is the number of K-means groups, and $.*$ means element-wise multiplying of vector. Unlike the traditional helper task design, which requires some experience, we employ clustered groups to establish the auxiliary tasks. Poor design of an auxiliary task can lead to negative knowledge transfer and affect the efficiency of the main task. Therefore, a benefit of our adaptive auxiliary task design is that we use the homogeneous fitness evaluation function and add a group mask based on physical location for each auxiliary task. The number of auxiliary tasks is determined by the number of groups.

B. Adaptive Auxiliary Task Selection

The multifactorial evolutionary process of MFMA-AAT is similar to classical MFEAs, while the first task in MFEA is the main task, the second task in MFEA is an active auxiliary task. In each generation, we select an auxiliary task among κ auxiliary tasks as the active auxiliary task. Only the active auxiliary participates in the evaluation to prevent too much computation. In the first generation, each auxiliary task is equally assigned a selection probability $\{\epsilon_k{=}1/\kappa | \epsilon_1, \epsilon_2, ..., \epsilon_k, ..., \epsilon_\kappa\}$. After each generation, MFMA-AAT records the optimal fitness value, namely, $\mathbb{L}(\mathbf{x}^\star)$, where $\mathbf{x}^\star$ is the individual with the lowest fitness value in population $\mathbf{X}$. If an auxiliary task has a higher drop rate in the fitness value compared with the previous generation, then the auxiliary task is given a greater active auxiliary selection probability in the next generation and vice versa. In addition, the constraint of selection probability for all auxiliary tasks is $\sum_1^\kappa \epsilon_k{=}1$. The drop rate δ of an auxiliary task in the n-th iteration is defined as follows:

$$\delta_k(n) = \frac{|\mathbb{L}(\mathbf{X}_n^\star) - \mathbb{L}(\mathbf{X}_{n-1}^\star)|}{\mathbb{L}(\mathbf{X}_{n-1}^\star)} \tag{10}$$

where $\mathbb{L}(\mathbf{X}_n^\star)$ indicates the lowest fitness value in the population of the n-th iteration. If k equals the index of activated auxiliary task in the n-th generation, and δ_k indicates a positive

improvement, the selection probability ϵ_k is increased by the percentage of δ_k, while selection probabilities of other inactivated auxiliary tasks are decreased on average to keep $\sum_1^\kappa \epsilon_k$=1, and vice versa. Hence, the active auxiliary task is adjusted with the selection probability dynamically.

C. Evolutionary Operators

In our multifactorial evolutionary process, we use a uniform crossover operator to randomly exchange selected genes between two parent chromosomes. The used mutation operator is a swap mutation that randomly selects two genes on a chromosome and interchanges their values, promoting diversity in the population and avoiding local optimum. Elitist selection strategy is employed for both parent and offspring populations.

Algorithm 2 Memetic method

Input: Population $\mathbf{X}$, population size λ, number of UE η, number of MEC servers ω, local search probability P_l
Output: Optimized population $\mathbf{X}'$

for $v = 1$ to $\lambda * P_l$ **do**
  for $i = 1$ to η **do**
    for $j = 1$ to ω **do**
      ξ^*_{ij} = ideal(P_i^j)
      $sim[j]$ = Similarity(ξ_{ij}, ξ^*_{ij})
      if $sim[j] > sim\_threshold$ **then**
        Φ_i.add(M_j)
      end if
    end for
  end for
  current_fitness $\leftarrow \mathbb{L}(\mathbf{X}_v)$
  optimal_fitness $\leftarrow$ current_fitness
  start_point $\leftarrow$ random_select(Φ_i)
  for M_j in Φ_i **do**
    $\mathbf{X}_v[i][j] \leftarrow M_j$
    val = $\mathbb{L}(\mathbf{X}_v^{'})$
    if val $<$ optimal_fitness **then**
      optimal_fitness $\leftarrow$ val
      $\mathbf{X}_v \leftarrow \mathbf{X}_v^{'}$
    end if
  end for
end for

1) Uniform Crossover: Given a pair of parent chromosomes $\mathbf{x}_v$ and $\mathbf{x}_u$ in the population,

$$\mathbf{x}_v : [\mathbf{c}_v^1, \mathbf{c}_v^2, \mathbf{c}_v^3, ..., \mathbf{c}_v^{i-1}, \mathbf{c}_v^i, ..., \mathbf{c}_v^\eta],$$
$$\mathbf{x}_u : [\mathbf{c}_u^1, \mathbf{c}_u^2, \mathbf{c}_u^3, ..., \mathbf{c}_u^{i-1}, \mathbf{c}_u^i, ..., \mathbf{c}_u^\eta],$$

the uniform crossover operation loops over each position i in the chromosome, and generates a random probability to judge whether $\mathbf{c}_v^i$ and $\mathbf{c}_u^i$ are to be exchanged. Compared with single-point or two-point crossover, the uniform crossover has more fine-grained exchange. The new pair of chromosomes is described as follows:

$$\mathbf{x}'_v : [\mathbf{c}_v^1, \mathbf{c}_u^2, \mathbf{c}_v^3, ..., \mathbf{c}_u^{i-1}, \mathbf{c}_v^i, ..., \mathbf{c}_u^\eta]$$
$$\mathbf{x}'_u : [\mathbf{c}_u^1, \mathbf{c}_v^2, \mathbf{c}_u^3, ..., \mathbf{c}_v^{i-1}, \mathbf{c}_u^i, ..., \mathbf{c}_v^\eta].$$

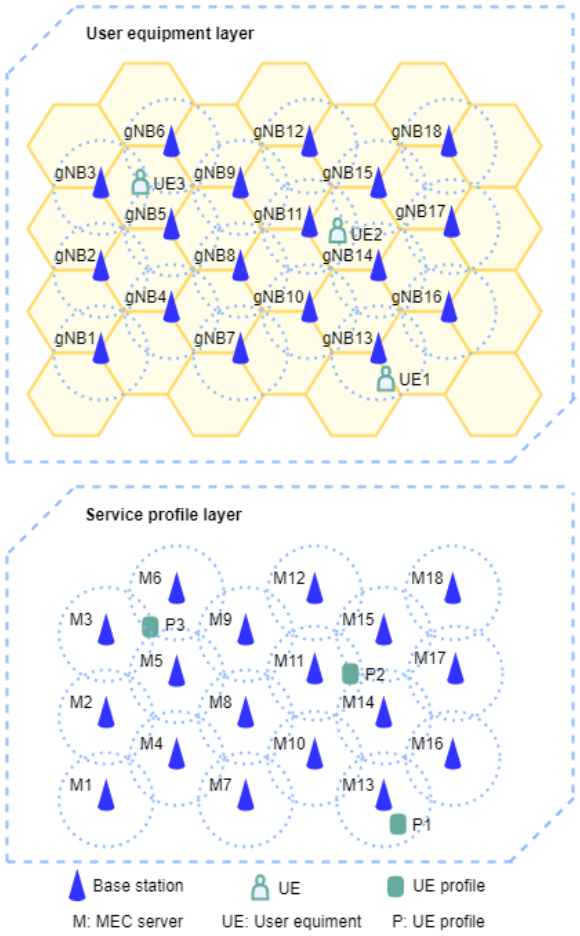


Fig. 5. MEC network layers. In a cellular network, the UE layer is physical, and the service profile layer is virtual. Each UE has a service profile only on an MEC server.

2) Swap Mutation: Given a chromosome, the swap mutation operation selects two random gene positions and then swaps them. For example, if the select mutation positions are i and j in a chromosome $\mathbf{x}$, i.e.,

$$\mathbf{x}_v : [\mathbf{c}_v^1, \mathbf{c}_v^2, \mathbf{c}_v^i, ..., \mathbf{c}_v^j, ..., \mathbf{c}_v^\eta],$$

the values of $\mathbf{x}_v^i$ and $\mathbf{x}_v^j$ are swapped:

$$\mathbf{x}'_v : [\mathbf{c}_v^1, \mathbf{c}_v^2, \mathbf{c}_v^j, ..., \mathbf{c}_v^i, ..., \mathbf{c}_v^\eta].$$

Unlike conventional single-point or multi-point mutation, swap mutation changes the ordering of genes instead of changing the content of genes, which results in smaller disturbance of the coding characteristics in the MEC environment.

3) Elitist Selection: The selection operator adopts the elitism strategy with linear rank selection. The chromosomes from both the parent and offspring populations are sorted according to their fitness values, and the best chromosome with the lowest user-perceived latency is first maintained. Then, the top-ranked chromosomes have a higher probability of selection without repetition.

D. Memetic Operator

MFMA-AAT adopts community detection that combines the physical features of MEC servers to reduce the combinatorial

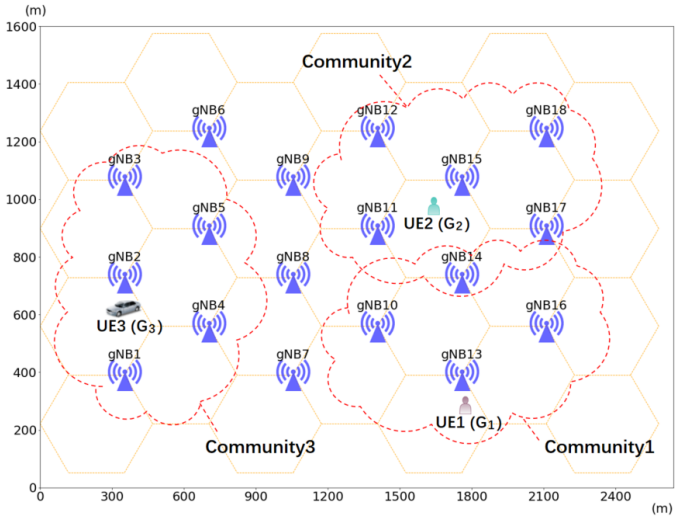


Fig. 6. Community detection. A UE profile can be placed on a subgroup of MEC servers representing a community that can provide sufficient computing capacity.

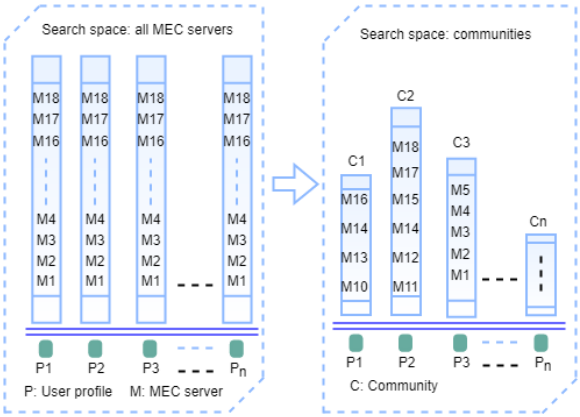


Fig. 7. The memetic method. Each randomly generated start point is used to establish a search within its corresponding community.

space, and subsequently searches in the neighborhood using hill climbing. The outline of memetic search is sumarized in Alg. 2.

1) Community Detection: In an MEC network, the placement of a UE profile on an edge server depends on the UE location, server computing resources, and various other factors. Fig. 5 shows a cellular network divided into virtual service profile layer and physical UE layer. The goal is to minimize user-perceived latency, and the profile is often placed on the closest edge server to the user. However, the real-world multi-application environment is complex, and various factors like UE location, server congestion, computing capacity, user priority, and type of edge server influence the user-perceived latency. To address this, we adopt community detection method based on cosine similarity to identify suitable MEC servers for profile placement based on critical attributes. The community detection method compares critical attributes of every M_j with an ideal placement. These suitable MEC servers form a community Φ_i for U_i, serving as a basis for local search optimization. The critical attributes of U_i in MEC server M_j are physical location ξ_{ij}^1, server congestion ξ_{ij}^2, and computing capacity ξ_{ij}^3:

$$\begin{aligned}\xi_{ij}^1 &= \mathrm{Hop}(\,\mathrm{IdxU}(U_i,t),\ \mathrm{IdxP}(P_i,t)\,),\\ \xi_{ij}^2 &= R_i^j + Q_i^j,\\ \xi_{ij}^3 &= \rho_j/\mu_j,\end{aligned} \tag{11}$$

where $\mathrm{Hop}(\,\mathrm{IdxU}(U_i,t),\ \mathrm{IdxP}(P_i,t)\,)$ is the hop distance from the MEC server that belongs to the UE-located base station to the profile-placed MEC server, and $R_i^j+Q_i^j$ is the calculation of requires and queues of P_i at M_j, ρ_j/μ_j is the computation capability of M_j. We use a vector $\xi_{ij}=(\xi_{ij}^1,\xi_{ij}^2,\xi_{ij}^3)$ to denote the critical attributes for profile P_i at MEC server M_j. The ideal placement achieves the optimal performance for each attribute. The values of optimal vector $\xi_{ij}^\star$ are the lowest hop distance, the lowest congestion, and the highest computing capacity that profile P_i can obtain in the MEC system. However, The MEC system is hard to provide ideal resources. Therefore, we tend to seek for a candidate MEC server with high similarity to the ideal placement $\xi_{ij}^\star$. The computation of cosine similarity between ξ_{ij} and $\xi_{ij}^\star$ is given by:

$$Similarity(\xi_{ij},\xi_{ij}^\star) = \frac{\sum_{\iota=1}^{3}\left(\xi_{ij}^\iota * \xi_{ij}^{\star\iota}\right)}{\sqrt{\sum_{\iota=1}^{3}(\xi_{ij}^\iota)^2} * \sqrt{\sum_{\iota=1}^{3}(\xi_{ij}^{\star\iota})^2}} \tag{12}$$

The similarity threshold in Alg. 2 can be set to 0.9. The samples of community detection are illustrated in Fig. 6.

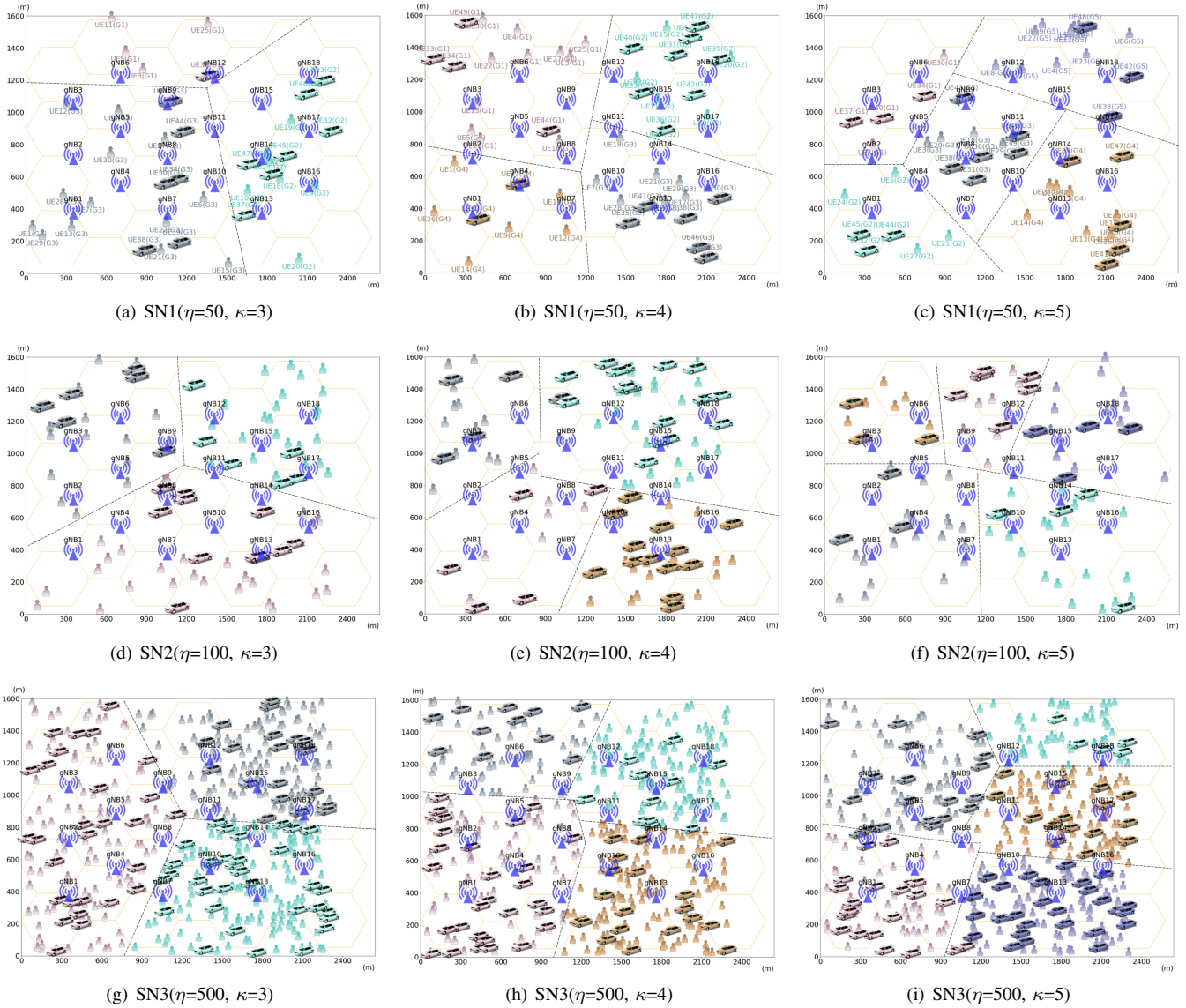
2) Local Search: Local search leverages hill climbing and community detection to enhance the search efficiency. As depicted in Fig. 7, during time slot t, user-specific profiles are placed in suitable MEC servers by detecting communities based on similarity comparisons of attributes. In other words, the search scope for profile placement per user is narrowed down from all MEC servers to the community set, which can overlap without causing conflicts. The candidate MEC servers in a community are ranked by similarity to facilitate the hill climbing process.

In the memetic method, the $\lfloor \eta * P_l \rfloor$ top-ranked chromosomes are selected for local search. The neighborhood search space is generated with the detected communities based on each chromosome. Position i is randomly selected on the chromosome, and the gene is replaced with other MEC server in the corresponding community. If the number of candidate servers in a community cannot reach the neighborhood size, the next gene is considered for replacement. For an individual

$$[M_{10}, M_{11}, M_1, ..., M_\eta],$$

if the initial selected position i is 1, then the neighborhood is given by:

$$\begin{gathered}[\boldsymbol{M_{13}}, M_{11}, M_1, ..., M_\eta]\\ [\boldsymbol{M_{14}}, M_{11}, M_1, ..., M_\eta]\\ [\boldsymbol{M_{16}}, M_{11}, M_1, ..., M_\eta]\\ [M_{10}, \boldsymbol{M_{12}}, M_1, ..., M_\eta]\\ [M_{10}, \boldsymbol{M_{14}}, M_1, ..., M_\eta]\\ [M_{10}, \boldsymbol{M_{15}}, M_1, ..., M_\eta]\\ ...\end{gathered}$$



(a) SN1(η=50, κ=3) (b) SN1(η=50, κ=4) (c) SN1(η=50, κ=5)

(d) SN2(η=100, κ=3) (e) SN2(η=100, κ=4) (f) SN2(η=100, κ=5)

(g) SN3(η=500, κ=3) (h) SN3(η=500, κ=4) (i) SN3(η=500, κ=5)

Fig. 8. Clustering results of UEs with $\kappa = \{3, 4, 5\}$ in different networks. The configurations of networks SN1-3 are summarized in Table III. η is the number of UEs and κ is the number of K-means groups.

The neighborhood size is usually set as $\lfloor 1/P_l \rfloor$. If the next chromosome in the neighborhood has an optimal fitness value, the climbing search continues until it encounters a worse value.

E. Termination Condition

In this study, we use three termination criteria: 1) the conventional maximum generation (Max^G) of EAs, 2) the maximum fitness evaluation (Max^E) of MFEAs, and 3) the maximum computing time (Max^T) of MFMAs, which is especially fair for fitness functions with high time cost.

The conventional EAs have a fixed number of fitness function evaluations per generation, whereas memetic algorithms vary in the number of fitness function evaluations as hill climbing performs local search without a fixed count. Furthermore, the cumulative time of memetic operation in each generation is also important. To accurately reflect the search efficiency of the memetic algorithm and fairly compare the evolutionary algorithms, we include Max^T in our termination criteria. If Max^T is reached, the iteration terminates in the current generation, even if Max^E or Max^G has not yet been reached.

F. Time Complexity

In computational time cost, MFMA-AAT boasts K-means clustering and memetic search over MFEAs. The time complexity of the K-means clustering algorithm is $O(\eta\kappa\tau)$, where η is the number of UEs, κ is the number of clusters (typically in the range of 3 to 5), and τ is the number of iterations in the clustering process. In practice, with $\eta = 500$, the average value of τ is 20 times, so both κ and τ can be considered as constants, thus yielding a time complexity of $O(\eta)$. Additionally, K-means clustering is only executed once during chromosome initialization, taking only a few hundred milliseconds and having a minimal impact on overall time.

The time complexity of memetic search, based on community structure, is $O(\eta\omega Max^G)$, where η is the number of UEs and ω is the number of servers. It involves computing the similarity for each server and selecting a group of suitable servers to reduce the search space. Given that $\omega = 18$ was a constant in our experiments, the actual time complexity of the memetic algorithm is $O(\eta Max^G)$. The time complexity of the multifactorial evolution is linearly correlated with the number of fitness evaluations, $O(Max^E)$, with either the main task or the auxiliary task counting as a fitness calculation. Max^E is generally set to $\lambda * Max^G$, where λ is the population size defined as a constant, so the time complexity of MFMA-AAT is $O(\eta Max^E)$.

IV. Experimental Studies

TABLE III
PARAMETERS OF MEC NETWORK

Network	Number of UE	Mobility level	Congestion level
SN1	50	$S1$	$C1$
SN2	100	$S2$	$C2$
SN3	500	$S3$	$C3$

TABLE IV
PARAMETER SETTINGS OF MFMA-ACH

Parameter	Description	Value
P_c	Probability of crossover	0.8
P_m	Probability of mutation	0.1
P_l	Probability of local search	0.5
λ	Size of the population	100
Max^G	The maximum number of generations	1000
Max^E	Maximum number of fitness evaluations	100,000

A. Test Problems

To study the performance of MEC systems, we use a general cellular network model consisting of ω base stations and η randomly positioned UEs. The network was generated using the wireless network simulation platform [41] that allows for customization of the base stations and UEs as shown in Fig. 8. In our experiment, the cellular network contains $\omega = 18$ MEC servers, as depicted in Fig. 2. Both the movement speed and the number of UEs in the MEC system could affect the average user-perceived latency, so we considered several simulated network scenarios. In Table III, the mobility level refers to the movement speed, while the congestion level denotes the level of crowdedness of an MEC server in a time slot. We experiment with three different UE counts of 50, 100, and 500. For the mobility level, we adjust the average movement speed by setting different speeds and scales for pedestrians and vehicles, resulting in three levels from $S1$ to $S3$, and the average speed is getting faster. For congestion level, we also have three levels from $C1$ to $C3$, and the congestion level is getting heavier. Hence, $SN1$ represents a network with few UEs and light congestion, $SN2$ is a network with normal traffic flow of pedestrians and vehicles, and $SN3$ is a highly congested network with many UEs assigned to certain MEC servers. The experimental hardware platform is equipped with an $i7-1255U$ $(1.7Ghz)$ processor and $16GB$ of RAM.

B. Parameter Settings

MEC networks face tradeoffs between the settings of various parameters [42]. For evolutionary operators, we compared different strategies of crossover including single-point, two-point, multi-point, and uniform crossover. In the mutation operator, we considered single-point, uniform, and swap mutation operators. The average convergence traces of MFMA-AAT over 20 runs using different crossover strategies on the networks SN1-SN3 with representative cluster number $\kappa = 5$ are shown in Fig. 9(a). Uniform crossover is shown to perform better than the other two strategies. The results of MFMA-AAT using different mutation operators are shown in Fig. 9(b), where single-point mutation and swap mutation outperform the other two methods, and swap mutation is slightly better than single-point mutation. Considering the overall performance, we adopted uniform crossover and swap mutation in MFMA-AAT.

To optimize the performance MFMA-AAT, we also fine-tuned its critical parameters, including the crossover probability P_c, the mutations probability P_m, and the memetic search probability P_l determines the probability of a chromosome undergoing memetic search. Therefore, we conducted trail-and-error evaluations of each parameter to determine the optimal settings for MFMA-AAT in the MEC network experiments.

To tune P_c, we first fixed other key parameters as (P_m=0.0, P_l=0.0, λ=100, Max^E=100,000) and then tested the values of P_c from 0.1 to 0.9 increasing by 0.1. The comparison results of using different P_c values are reported in Table V. The significance of performance difference by using different values is test with Friedman test and the significance of performance difference between the best setting and each other settings is test with t-test. The corresponding test p-values are also reported in Table V. It can be seen that with $P_c \geq 0.6$, MFMA-AAT obtains better performance and the optimal setting is $P_c = 0.8$. We conducted similar tuning experiments on other parameters. For P_m, the test results are listed in Table VI with (P_c=0.8, P_l=0.0, λ=100, Max^E=100,000), and P_m ranging from 0.1 to 0.9. At mutation probability P_m of 0.1, MFMA-AAT achieved the best performance. The memetic operator performs a local search for a chromosome in its neighborhood. Probability P_l is used to control the proportion of chromosomes for memetic search. We tested P_l from 0.1 to 0.9 and fixed other parameters (P_c=0.0, P_m=0.0, λ=100, Max^E=100,000) to suppress their interference. The test results are listed in Table VII. The optimal setting of P_l for MFMA-AAT is obtained at 0.5.

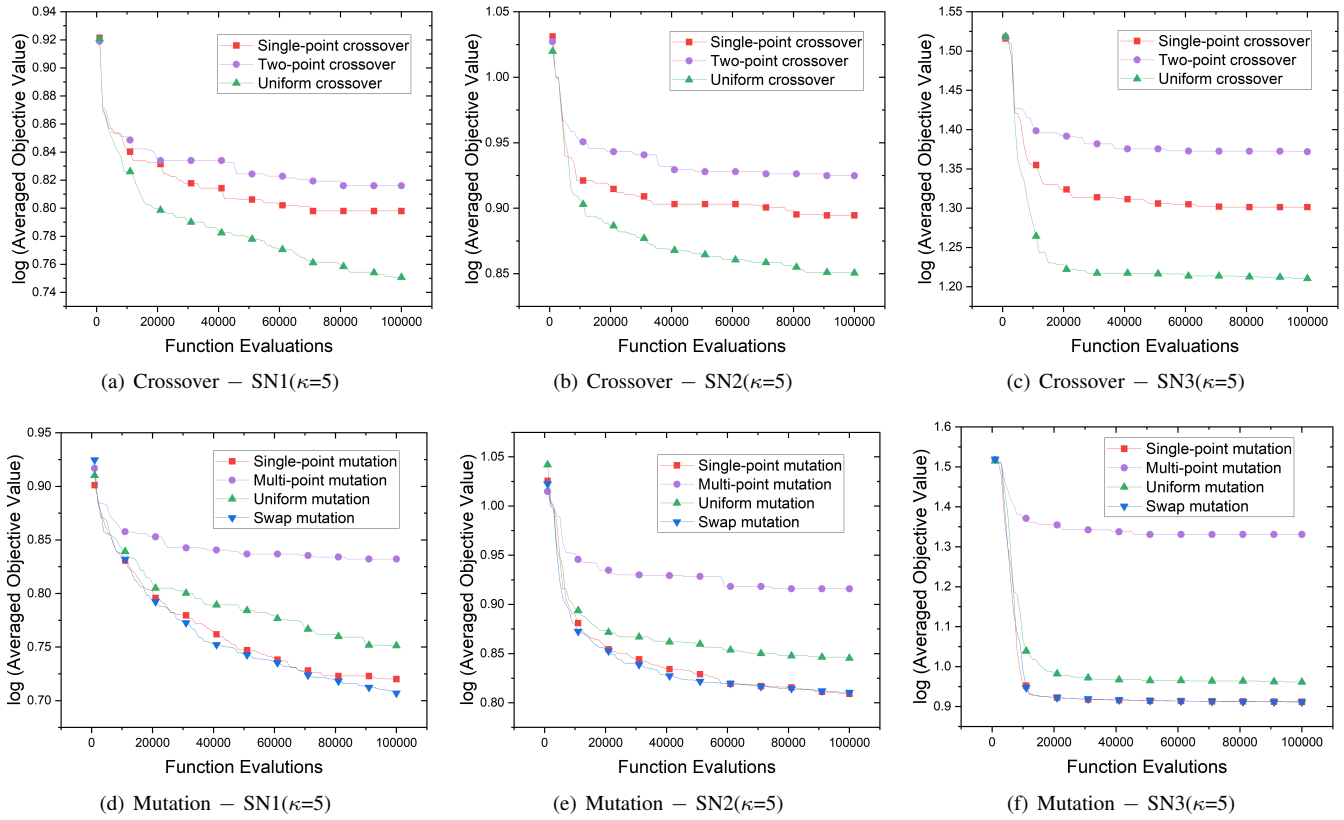
The above tests all run over 20 independent trials then we obtained the optimal parameter settings listed in Table IV.

C. Comparison Results

To evaluate the proposed MFMA-AAT on user-perceived latency, we compared it with traditional MEC algorithm GT [12], conventional genetic algorithm (GA) [43], the state-of-the-art MFEAs including MFEA-II [18] and MFEA-AKT* [32], and MFEA-AAT, which is MFMA-AAT without using memetic operator. In MFEA-II, two optimization tasks, i.e., L defined in Eq.(7) and L^α defined in Eq.(3) were considered.

TABLE V
PARAMETERS COMPARISON OF CROSSOVER OVER 20 INDEPENDENT TRIALS

Data	P_c=0.1	P_c=0.2	P_c=0.3	P_c=0.4	P_c=0.5	P_c=0.6	P_c=0.7	P_c=0.8	P_c=0.9	Friedman P-value
SN1(κ=3)	6.07±0.12(+)	6.09±0.04(+)	5.87±0.06(+)	5.95±0.09(+)	5.95±0.14(+)	6.1±0.07(+)	5.88±0.1(+)	**5.74±0.11** *	5.77±0.1(+)	3.00E-02
T-test P-value	4.00E-02	1.00E-02	2.30E-01	1.20E-01	1.70E-01	2.00E-02	2.50E-01	–	8.40E-01	
SN1(κ=4)	6.1±0.04(+)	5.87±0.12(+)	5.93±0.09(+)	5.89±0.09(+)	5.92±0.08(+)	5.98±0.04(+)	5.75±0.07(+)	**5.58±0.09** *	5.66±0.03(+)	2.00E-02
T-test P-value	1.70E-03	5.00E-02	2.00E-02	3.00E-02	2.00E-02	4.70E-03	1.10E-01	–	3.00E-01	
SN1(κ=5)	6.08±0.05(+)	6.05±0.04(+)	6.17±0.0047(+)	5.96±0.06(+)	5.92±0.02(+)	6.01±0.02(+)	5.87±0.07(+)	5.76±0.09(+)	**5.7±0.07** *	5.20E-03
T-test P-value	3.70E-03	3.50E-03	7.00E-04	2.00E-02	1.00E-02	3.70E-03	7.00E-02	5.10E-01	–	
SN2(κ=3)	8.0±0.01(+)	8.02±0.01(+)	7.94±0.08(+)	8.0±0.02(+)	7.83±0.08(+)	7.86±0.09(+)	7.87±0.04(+)	**7.7±0.09** *	7.82±0.05(+)	1.00E-02
T-test P-value	1.00E-02	9.00E-03	5.00E-02	1.00E-02	2.30E-01	1.60E-01	8.00E-02	–	2.00E-01	
SN2(κ=4)	7.89±0.04(+)	7.88±0.06(+)	7.8±0.07(+)	7.81±0.05(+)	7.74±0.06(+)	7.79±0.05(+)	7.76±0.04(+)	7.62±0.1(+)	**7.61±0.04** *	2.00E-02
T-test P-value	2.10E-03	7.60E-03	3.00E-02	1.00E-02	7.00E-02	2.00E-02	3.00E-02	9.70E-01	–	
SN2(κ=5)	7.46±0.09(+)	7.48±0.1(+)	7.45±0.07(+)	7.5±0.04(+)	7.45±0.06(+)	7.43±0.07(+)	7.35±0.04(+)	**7.29±0.01** *	7.29±0.02(+)	5.00E-02
T-test P-value	6.00E-02	5.00E-02	3.00E-02	1.60E-03	2.00E-02	4.00E-02	7.00E-02	–	7.00E-01	
SN3(κ=3)	14.44±0.08(+)	14.16±0.04(+)	13.72±0.18(+)	13.67±0.13(+)	13.44±0.32(+)	13.29±0.23(+)	12.78±0.61(+)	**12.33±0.3** *	12.91±0.04(+)	6.20E-03
T-test P-value	6.00E-04	1.00E-03	4.90E-03	4.20E-03	2.00E-02	2.00E-02	4.00E-01	–	5.00E-02	
SN3(κ=4)	14.56±0.08(+)	14.37±0.03(+)	14.12±0.07(+)	13.79±0.15(+)	13.79±0.07(+)	13.81±0.08(+)	13.66±0.03(+)	**13.18±0.17** *	13.27±0.07(+)	3.80E-03
T-test P-value	5.00E-04	7.00E-04	2.10E-03	2.00E-02	1.00E-02	1.00E-02	2.00E-02	–	5.50E-01	
SN3(κ=5)	18.15±0.08(+)	18.12±0.07(+)	17.95±0.14(+)	17.84±0.06(+)	17.48±0.1(+)	17.65±0.06(+)	17.56±0.08(+)	**16.93±0.17** *	17.15±0.17(+)	4.20E-03
T-test P-value	7.00E-04	8.00E-04	2.50E-03	1.90E-03	2.00E-02	4.50E-03	8.70E-03	–	2.60E-01	
*	0	0	0	0	0	0	0	**7**	2	



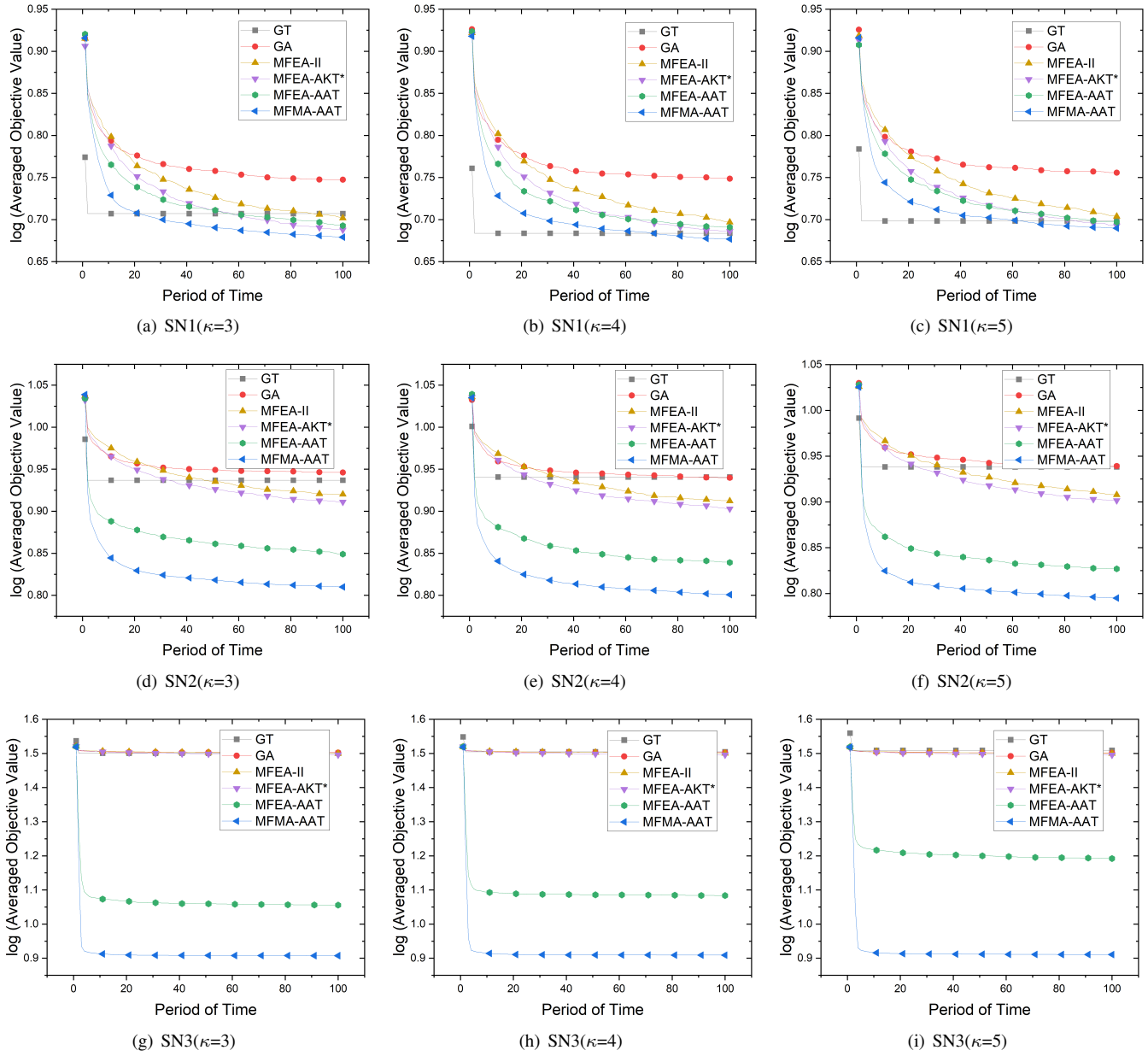
(a) Crossover – SN1(κ=5) (b) Crossover – SN2(κ=5) (c) Crossover – SN3(κ=5)

(d) Mutation – SN1(κ=5) (e) Mutation – SN2(κ=5) (f) Mutation – SN3(κ=5)

Fig. 9. Average convergence traces using different crossover and mutation operators on SN1-3 networks ($\kappa = 5$) over 20 runs.

Here L^{α} is empirically set as the auxiliary task as it is the most similar subtask to the main task L. MFEA-AKT* is modified from the original MFEA-AKT [32] that dynamically selects different crossover operators to control the efficiency of knowledge transfer in each generation. Instead of selecting crossover, in MFEA-AKT* we applied the same selection strategy to select auxiliary tasks from $L_i^{\alpha}(t)$ (Eq.(3)), $L_i^{\beta}(t)$ (Eq.(4)), and $L_i^{\gamma}(t)$ (Eq.(6)). Note that since there are no MFEAs implementing similar auxiliary task selection mechanism in the literature, the comparison to MFEA-II and MFEA-AKT* with the current configurations might not be absolutely fair. Nevertheless, it could reflect the performance of the proposed MFMA-AAT to some extent.

In the comparison experiments, each memetic search operator takes approximately a few hundred milliseconds. By setting Max^G=1000, even if we stop at 100,000 fitness calls, the time overhead of all memetic search operators is still significantly higher than that of MFEA, thus, we stopped all algorithms at the same maximum running time to ensure a fair comparison. When drawing the convergence trace in Fig. 10, 100 points are taken evenly in running time span, and each point on the horizontal axis denotes 0.25, 0.4 and 1.5 second for $SN1$, $SN2$ and $SN3$ data, respectively. Table VIII summarizes the average objective value comparison of all algorithms, and Fig. 10 shows the evolutionary process according to running time in three different networks with



(a) SN1(κ=3) (b) SN1(κ=4) (c) SN1(κ=5)

(d) SN2(κ=3) (e) SN2(κ=4) (f) SN2(κ=5)

(g) SN3(κ=3) (h) SN3(κ=4) (i) SN3(κ=5)

Fig. 10. Convergence trace of the compared algorithms on different networks with different cluster numbers.

cluster number $\kappa = \{3, 4, 5\}$. The significance of performance difference among all algorithms is test with Friedman test and the significance of performance difference between MFMA-AAT and each other algorithms is test with t-test. The corresponding test p-values are also reported in Table VIII. As the number of UEs and network load increased from $SN1$ to $SN3$, the user-perceived latency result also increased, yet MFMA-AAT always managed to achieve the significantly best result in the test networks.

In the case the amount of UE is small, i.e., $SN1$, the multi-tasking MFEAs outperform the single-task method GA, and the difference between the MFEAs is small. MFMA-AAT shows slight superiority to other methods. GT does not start from a random initial value like EAs, so the initial fitness value of GT is lower than EAs. GT converges very fast but mostly to local optima. Since GT is a deterministic method, it was not included in the significance test with other methods in Table VIII. As the amount of UE increases, such as in $SN2$, the search space becomes larger, GT is more likely get trapped into local optima. MFEAs can jump out of the local optima to detect a lower fitness value. All MFEAs outperform GT and GA thanks to the used of auxiliary tasks. MFEA-AKT* performs better than MFEA-II in SN1 and SN2 which suggests the dynamic selection of auxiliary tasks provides more positive knowledge transfer than using a constant auxiliary task. MFEA-AKT*, MFEA-AAT and MFMA-AAT all update auxiliary tasks dynamically, but the constructions of auxiliary tasks are different. MFEA-AAT and MFMA-AAT achieve faster convergence and detect lower fitness values because in these two algorithms the

TABLE VI
PARAMETERS COMPARISON OF MUTATION OVER 20 INDEPENDENT TRIALS

Data	P_m=0.1	P_m=0.2	P_m=0.3	P_m=0.4	Friedman
SN1(κ=3)	**5.49±0.09***	5.95±0.08(+)	6.23±0.07(+)	6.43±0.09(+)	8.91E-29
T-test P-value	–	5.77E-19	1.09E-26	8.64E-29	
SN1(κ=4)	**5.44±0.08***	5.93±0.08(+)	6.25±0.08(+)	6.44±0.11(+)	3.33E-28
T-test P-value	–	1.09E-20	1.51E-28	2.27E-29	
SN1(κ=5)	**5.6±0.09***	6.08±0.1(+)	6.4±0.08(+)	6.57±0.11(+)	1.40E-27
T-test P-value	–	1.30E-18	4.93E-28	5.31E-28	
SN2(κ=3)	**7.24±0.05***	7.59±0.07(+)	7.87±0.07(+)	8.11±0.07(+)	1.81E-30
T-test P-value	–	1.20E-20	1.79E-28	1.51E-33	
SN2(κ=4)	**7.12±0.08***	7.49±0.07(+)	7.78±0.07(+)	8.05±0.08(+)	2.31E-30
T-test P-value	–	1.30E-17	1.86E-26	1.74E-30	
SN2(κ=5)	**7.03±0.06***	7.42±0.09(+)	7.78±0.1(+)	8.13±0.1(+)	2.40E-30
T-test P-value	–	1.24E-18	7.41E-28	1.17E-33	
SN3(κ=3)	**8.63±0.05***	9.9±0.14(+)	12.22±0.15(+)	15.07±0.2(+)	2.78E-32
T-test P-value	–	7.41E-32	7.71E-48	1.08E-52	
SN3(κ=4)	**8.78±0.06***	10.63±0.16(+)	13.48±0.29(+)	16.77±0.31(+)	3.21E-32
T-test P-value	–	1.47E-35	1.57E-41	1.87E-49	
SN3(κ=5)	**9.17±0.07***	12.03±0.19(+)	15.62±0.26(+)	19.03±0.32(+)	3.60E-32
T-test P-value	–	4.82E-40	2.25E-48	4.10E-52	
*	**9**	0	0	0	

selected auxiliary tasks based on historical contributions are more efficient for knowledge transfer. Moveover, MFMA-AAT outperforms MFEA-AAT. The improvement of MFMA-AAT over MFEA-AAT is attributed to the addition of a memetic operator, which substantially improve the local convergence of the algorithm. Under high mobility at level $S3$ and heavy congestion at level $C3$ in $SN3$, the superiority of MFEA-AAT and MFMA-AAT to other methods is more significant.

The advantage of MFMA-AAT lies in the adaptive selection of auxiliary tasks in different locations and time periods. In real MEC scenarios, for example, shopping malls and office buildings are featured by high UE density, and the aggregation of UEs in such spaces is more obvious. In the crowded area, there is more competition for MEC computing resources, where service migration optimization is more desirable. The K-means clustering method can divide these areas, based on which we can design the corresponding auxiliary tasks. Moreover, the user mobility is the most important feature of MEC networks, thus the clustering groups of UEs must change over time. For example, UEs are highly aggregated in office buildings during the daytime on weekdays, but the aggregation disappears at night and on holidays. Whenever there is a change in the UE group of the MEC network, the algorithm should be able to capture the changes on time. Unlike the existing algorithms that use sub-tasks or similar tasks to construct auxiliary tasks, MFMA-AAT constructs auxiliary tasks based on the real-time physical characteristics of MEC, leading to more accurate optimization. In summary, MFMA-AAT has a faster convergence speed and can detect more promising service migration solutions. This is because MFMA-AAT dynamically constructs auxiliary tasks in each time slot and selects a more efficient auxiliary task with maximum positive knowledge transfer based on spatiotemporal changes of the MEC network.

V. CONCLUSION

5G and future mobile networks require ultralow user-perceived latency. An MEC has been introduced to address these challenges. Service migration optimization is a critical technology to achieve ultralow latencies of mobile UE in MEC systems. Recent studies have demonstrated the potential of EAs in combinatorial optimization problems. Moreover, community detection helps unveil non-trivial organization of a cellular network at the mesoscopic scale and can be applied to MEC.

We use MFEA paradigms for service migration in MEC. Specifically, we propose MFMA-AAT. MFMA-AAT uses K-means to cluster UE and design adaptive auxiliary tasks without expert knowledge according to cluster groups, and thereafter adopts adaptive selection to obtain an effective auxiliary task among subtasks. MFMA-AAT detects a community to decode a group of feasible MEC servers to efficiently place UE profiles. In a congested latency-sensitive MEC network, MFMA-AAT can achieve considerably lower user-perceived latency than traditional algorithms and recent MFEAs. Future research will be applied to energy-efficient scenarios. The source code for MFMA-AAT is available at https://github.com/basket163/MFMA-AAT.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China [61871272] and [62001300].

REFERENCES

[1] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for iov using 5g NR-V2X communication," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10 397–10 410, 2022.

[2] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 125:1–125:36, 2020.

[3] T. Wang, Y. Zhang, N. N. Xiong, S. Wan, S. Shen, and S. Huang, "An effective edge-intelligent service placement technology for 5g-and-beyond industrial iot," *IEEE Trans. Ind. Informatics*, vol. 18, no. 6, pp. 4148–4157, 2022.

[4] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3448–3459, 2021.

[5] Z. Zhou, H. Liao, B. Gu, S. Mumtaz, and J. Rodriguez, "Resource sharing and task offloading in iot fog computing: A contract-learning approach," *IEEE Trans. Emerging Topics in Comput. Intellig.*, vol. 4, no. 3, pp. 227–240, 2020.

[6] S. Xiao, C. Liu, K. Li, and K. Li, "System delay optimization for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 109, no. 8, pp. 17–28, 2020.

[7] K. Zhu, Y. Zhou, C. Yi, and R. Wang, "Computation resource configuration with adaptive qos requirements for vehicular edge computing: A fluid-model based approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 21 148–21 162, 2022.

[8] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[9] Y. Liu, X. Wang, G. Boudreau, A. B. Sediq, and H. Abou-zeid, "Deep learning based hotspot prediction and beam management for adaptive virtual small cell in 5g networks," *IEEE Trans. Emerging Topics in Comput. Intellig.*, vol. 4, no. 1, pp. 83–94, 2020.

[10] M. R. Anwar, S. Wang, M. F. Akram, S. Raza, and S. Mahmood, "5g-enabled MEC: A distributed traffic steering for seamless service migration of internet of vehicles," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 648–661, 2022.

[11] X. Li, S. Chen, Y. Zhou, J. Chen, and G. Feng, "Intelligent service migration based on hidden state inference for mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 380–393, 2022.

[12] H. Wang, T. Lv, Z. Lin, and J. Zeng, "Energy-delay minimization of task migration based on game theory in mec-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8175–8188, 2022.

[13] L. Sun, J. Wang, and B. Lin, "Task allocation strategy for mec-enabled iiots via bayesian network based evolutionary computation," *IEEE Trans. Ind. Informatics*, vol. 17, no. 5, pp. 3441–3449, 2021.

TABLE VII
PARAMETER COMPARISON OF LOCAL SEARCH OVER 20 INDEPENDENT TRIALS

Data	P_l=0.1	P_l=0.2	P_l=0.3	P_l=0.4	P_l=0.5	P_l=0.6	P_l=0.7	P_l=0.8	P_l=0.9	Friedman P-value
SN1(κ=3)	5.48±0.1(+)	5.45±0.08(+)	5.5±0.08(+)	5.38±0.12(+)	**5.37±0.09***	5.5±0.08(+)	5.51±0.08(+)	5.46±0.08(+)	5.47±0.07(+)	5.11E-05
T-test P-value	8.00E-04	1.00E-02	6.54E-05	9.50E-01	–	4.21E-05	1.65E-05	3.80E-03	4.00E-04	
SN1(κ=4)	5.43±0.1(+)	5.43±0.11(+)	5.49±0.08(+)	5.37±0.09(+)	**5.36±0.09***	5.41±0.08(+)	5.42±0.08(+)	5.43±0.09(+)	5.44±0.12(+)	1.00E-02
T-test P-value	5.00E-02	6.00E-02	6.62E-05	7.80E-01	–	1.30E-01	6.00E-02	3.00E-02	3.00E-02	
SN1(κ=5)	5.6±0.11(+)	5.6±0.08(+)	5.57±0.09(+)	5.48±0.08(+)	**5.44±0.1***	5.59±0.11(+)	5.58±0.1(+)	5.58±0.07(+)	5.56±0.09(+)	1.20E-07
T-test P-value	2.68E-05	5.16E-06	2.00E-04	1.70E-01	–	2.00E-04	1.00E-04	2.81E-05	5.00E-04	
SN2(κ=3)	7.25±0.06(+)	7.24±0.05(+)	7.24±0.08(+)	7.24±0.06(+)	**7.23±0.05***	7.26±0.05(+)	7.24±0.07(+)	7.24±0.05(+)	7.24±0.05(+)	7.30E-01
T-test P-value	3.30E-01	4.70E-01	6.30E-01	6.20E-01	–	1.30E-01	6.10E-01	8.30E-01	6.10E-01	
SN2(κ=4)	7.13±0.06(+)	7.12±0.06(+)	7.11±0.08(+)	7.13±0.03(+)	7.11±0.06(+)	7.12±0.08(+)	7.11±0.05(+)	**7.07±0.05***	7.09±0.05(+)	3.00E-02
T-test P-value	1.70E-03	8.90E-03	4.00E-02	7.16E-05	2.00E-02	1.00E-02	1.00E-02	–	2.10E-01	
SN2(κ=5)	6.98±0.06(+)	7.01±0.07(+)	7.0±0.06(+)	6.98±0.06(+)	6.98±0.05(+)	7.01±0.07(+)	**6.97±0.08***	6.97±0.06(+)	7.01±0.08(+)	2.10E-01
T-test P-value	6.30E-01	7.00E-02	1.90E-01	5.00E-01	4.80E-01	5.00E-02	–	8.60E-01	1.30E-01	
SN3(κ=3)	8.61±0.04(+)	8.63±0.04(+)	8.61±0.03(+)	8.61±0.03(+)	**8.6±0.05***	8.61±0.04(+)	8.63±0.04(+)	8.61±0.04(+)	8.62±0.04(+)	5.10E-01
T-test P-value	6.50E-01	1.50E-01	5.10E-01	6.00E-01	–	6.90E-01	7.00E-02	7.60E-01	2.80E-01	
SN3(κ=4)	8.78±0.05(+)	8.76±0.04(+)	8.79±0.06(+)	**8.73±0.07***	8.77±0.06(+)	8.77±0.04(+)	8.77±0.04(+)	8.78±0.06(+)	8.77±0.05(+)	5.60E-01
T-test P-value	1.00E-02	1.00E-01	1.00E-02	–	1.00E-01	3.00E-02	5.00E-02	4.00E-02	1.00E-01	
SN3(κ=5)	9.21±0.09(+)	9.17±0.08(+)	9.2±0.08(+)	9.22±0.05(+)	**9.16±0.08***	9.21±0.06(+)	9.19±0.08(+)	9.21±0.07(+)	9.23±0.06(+)	1.00E-01
T-test P-value	1.20E-01	9.20E-01	1.20E-01	9.30E-03	–	7.00E-02	2.60E-01	6.00E-02	1.00E-02	
*	0	0	0	1	**6**	0	1	1	0	

TABLE VIII
FITNESS VALUE COMPARISON OVER 20 INDEPENDENT TRIALS ON THREE NETWORKS, EACH NETWORK SETS THREE CLUSTERING NUMBERS

Data	**GT**	**GA**	**MFEA-II**	**MFEA-AKT***	**MFEA-AAT**	**MFMA-AAT**	Friedman P-value
SN1(κ=3)	5.09 ± 0.08(+)	5.59 ± 0.1(+)	5.03 ± 0.08(+)	4.87 ± 0.05(+)	4.93 ± 0.08(+)	**4.78 ± 0.09**	8.68E-14
T-test P-value	–	1.33E-26	2.15E-11	1.76E-4	1.14E-06	–	
SN1(κ=4)	4.82 ± 0.05(+)	5.61 ± 0.1(+)	4.98 ± 0.06(+)	4.85 ± 0.04(+)	4.91 ± 0.07(+)	**4.75 ± 0.09**	4.25E-14
T-test P-value	–	1.80E-26	4.07E-11	6.81E-05	3.97E-07	–	
SN1(κ=5)	4.99 ± 0.08(+)	5.69 ± 0.09(+)	5.06 ± 0.08(+)	4.93 ± 0.06(+)	4.98 ± 0.06(+)	**4.89 ± 0.09**	6.89E-13
T-test P-value	–	5.38E-26	1.62E-06	0.2134	2.91E-3	–	
SN2(κ=3)	8.65 ± 0.06(+)	8.83 ± 0.08(+)	8.32 ± 0.06(+)	8.15 ± 0.06(+)	7.06 ± 0.08(+)	**6.46 ± 0.05**	1.82E-19
T-test P-value	–	1.80E-49	6.87E-49	2.34E-46	4.76E-27	–	
SN2(κ=4)	8.72 ± 0.12(+)	8.71 ± 0.08(+)	8.17 ± 0.06(+)	7.99 ± 0.08(+)	6.90 ± 0.06(+)	**6.32 ± 0.05**	1.12E-19
T-test P-value	–	2.21E-49	4.14E-48	7.05E-44	1.82E-29	–	
SN2(κ=5)	8.68 ± 0.11(+)	8.69 ± 0.08(+)	8.09 ± 0.07(+)	7.97 ± 0.06(+)	6.71 ± 0.07(+)	**6.24 ± 0.07**	1.19E-18
T-test P-value	–	1.83E-47	1.03E-44	7.78E-45	7.85E-23	–	
SN3(κ=3)	31.66 ± 0.22(+)	31.79 ± 0.05(+)	31.61 ± 0.07(+)	31.35 ± 0.03(+)	11.36 ± 0.45(+)	**8.08 ± 0.02**	1.67E-19
T-test P-value	–	2.22E-95	4.76E-92	5.16E-102	7.34E-29	–	
SN3(κ=4)	31.96 ± 0.22(+)	31.79 ± 0.05(+)	31.65 ± 0.05(+)	31.34 ± 0.04(+)	12.12 ± 0.08(+)	**8.11 ± 0.03**	2.37E-19
T-test P-value	–	1.04E-95	1.21E-95	3.60E-98	4.98E-59	–	
SN3(κ=5)	32.27 ± 0.22(+)	31.71 ± 0.05(+)	31.61 ± 0.05(+)	31.32 ± 0.05(+)	15.57 ± 0.16(+)	**8.14 ± 0.02**	2.49E-19
T-test P-value	–	1.04E-95	1.21E-95	3.60E-98	4.98E-59	–	

[14] A. Gupta, Y. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, 2016.

[15] T. Wei, S. Wang, J. Zhong, D. Liu, and J. Zhang, "A review on evolutionary multitask optimization: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 941–960, 2022.

[16] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2018.

[17] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization - a new frontier in evolutionary computation research," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 22–33, 2021.

[18] K. K. Bali, Y. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 69–83, 2020.

[19] X. Ma, J. Yin, A. Zhu, X. Li, Y. Yu, L. Wang, Y. Qi, and Z. Zhu, "Enhanced multifactorial evolutionary algorithm with meme helper-tasks," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 7837–7851, 2022.

[20] Q. Shang, Y. Huang, Y. Wang, M. Li, and L. Feng, "Solving vehicle routing problem by memetic search with evolutionary multitasking," *Memetic Comput.*, vol. 14, no. 1, pp. 31–44, 2022.

[21] C. Yang, J. Ding, Y. Jin, C. Wang, and T. Chai, "Multitasking multi-objective evolutionary operational indices optimization of beneficiation processes," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1046–1057, 2019.

[22] S. Das and A. Biswas, "Deployment of information diffusion for community detection in online social networks: A comprehensive review," *IEEE Trans. Comput. Soc. Syst.*, vol. 8, no. 5, pp. 1083–1107, 2021.

[23] A. Zhu and Y. Wen, "Computing offloading strategy using improved genetic algorithm in mobile edge computing system," *J. Grid Comput.*, vol. 19, no. 3, p. 38, 2021.

[24] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, 2021.

[25] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8805–8819, 2020.

[26] J. Liu, C. Liu, B. Wang, G. Gao, and S. Wang, "Optimized task allocation for iot application in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10 370–10 381, 2022.

[27] J. Wang, K. Zhu, B. Chen, and Z. Han, "Distributed clustering-based cooperative vehicular edge computing for real-time offloading requests," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 653–669, 2022.

[28] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Ai-based network-aware service function chain migration in 5g and beyond networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 472–484, 2022.

[29] H. Tang, H. Wu, Y. Zhao, and R. Li, "Joint computation offloading and resource allocation under task-overflowed situations in mobile-edge computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1539–1553, 2022.

[30] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, and F. Zeng, "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052,

2020.
[31] T. Wei, S. Wang, J. Zhong, D. Liu, and J. Zhang, "A review on evolutionary multitask optimization: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 941–960, 2022.
[32] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2563–2576, 2021.
[33] H. Xu, A. K. Qin, and S. Xia, "Evolutionary multitask optimization with adaptive knowledge transfer," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 290–303, 2022.
[34] 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 Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 7, pp. 4457–4469, 2022.
[35] K. Wu, C. Wang, and J. Liu, "Evolutionary multitasking multilayer network reconstruction," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 12 854–12 868, 2022.
[36] A. D. Martinez, J. D. Ser, E. Osaba, and F. Herrera, "Adaptive multifactorial evolutionary optimization for multitask reinforcement learning," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 233–247, 2022.
[37] L. Feng, Y. Huang, L. Zhou, J. Zhong, A. Gupta, K. Tang, and K. C. Tan, "Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle routing problem," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3143–3156, 2021.
[38] G. Li, Q. Zhang, and Z. Wang, "Evolutionary competitive multitasking optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 278–289, 2022.
[39] H. Wang, L. Feng, Y. Jin, and J. Doherty, "Surrogate-assisted evolutionary multitasking for expensive minimax optimization in multiple scenarios," *IEEE Comput. Intell. Mag.*, vol. 16, no. 1, pp. 34–48, 2021.
[40] S. Liu, H. Wang, W. Peng, and W. Yao, "A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1087–1101, 2022.
[41] E. D. Santis, A. Giuseppi, A. Pietrabissa, M. Capponi, and F. D. Priscoli, "Satellite integration into 5g: Deep reinforcement learning for network selection," *Int. J. Autom. Comput.*, vol. 19, no. 2, pp. 127–137, 2022.
[42] G. Li, L. Liu, Z. Liang, X. Ma, and Z. Zhu, "Memetic algorithm based on community detection for energy-efficient service migration optimization in 5g mobile edge computing," in *32nd IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2021, Helsinki, Finland, September 13-16, 2021*. IEEE, 2021, pp. 1–7.
[43] J. H. Holland, *Adaptation in natural artificial systems, 2nd edition*. MIT Press, 1992.