Virtual Resource Allocation for Mobile Edge Computing: A Hypergraph Matching Approach

Long Zhang*, Hongliang Zhang†, Lisu Yu‡, Haitao Xu§, Lingyang Song†, and Zhu Han¶‖
*School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China
†School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
‡Institute of Mobile Communications, Southwest Jiaotong University, Chengdu 611756, China
§School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
¶Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA
‖Department of Computer Science and Engineering, Kyung Hee University, Seoul 02447, South Korea

***Abstract*—In this paper, the energy efficient virtual machine (VM) placement and virtual resource allocation problem for the mobile edge computing (MEC) system is explored. Particularly, we develop an optimization framework of energy consumption minimization for computing and offloading by jointly optimizing the VM placement matrix and the number of physical machines (PMs). To resolve this problem, we transform the optimization problem into a non-uniform weighted hypergraph model. In this model, the weight of hyperedge is defined as the negative of the accumulated energy consumption for computing at VM instances hosted by one PM. Based on the hypergraph model, a hypergraph matching algorithm by utilizing the local search policy is proposed for finding the maximum-weight subset of vertex-disjoint hyperedges, aiming to obtain an optimal VM placement, i.e., (M^*)-perfect matching. Furthermore, the virtualized resources are further allocated to user equipments (UEs) in the form of multiple VM instances via the optimal VM placement to meet the requirement of workloads. Simulation results are presented to demonstrate the effectiveness of the proposed hypergraph matching algorithm over the alternative benchmark algorithm.**

I. Introduction

With the technological advances in the Internet of Things and 5G, along with the rapid proliferation of massive internet-connected things, various computation-intensive and latency-sensitive applications have been emerging, such as ultra-high definition videos, augmented reality, cloud gaming, artificial intelligence, autonomous driving, etc. Mobile edge computing (MEC) has provided a promising solution to extend computation and storage resources to the network edge (e.g., cellular eNBs) in proximity to mobile user equipments (UEs) [1]–[3]. Instead of relying on a remote cloud, MEC liberates the resource-limited UEs from heavy computation workloads of emerging applications by enabling them to offload the computation tasks to closer computation servers hosted by micro MEC data centers (e.g., fog nodes and cloudlets). As such, it has potentials to bring multiple benefits including reduced latency, enhanced security, alleviated backhaul congestion, enhanced user experience, improved energy saving, etc.

In spite of being advantageous, different network services and third party applications for MEC make operators very difficult to manage, by identifying specific service for each physical device running on their networks. Recently, Network Functions Virtualization (NFV) has attracted intensive attention from both academia and industry as a key enabler for MEC, to virtualize network services that run on dedicated hardware devices, known as physical machines (PMs) [4]. By means of NFV, network functions can be packaged as multiple virtual machines (VMs) on top of commoditized PMs. Such virtualization enables deployment of new network services and elastic network scaling to reduce maintenance costs and make network more flexible, scalable and cost-effective. To implement the NFV-enabled MEC architecture, virtual network functions (e.g., virtualized computing and storage services) are deployed at the network edge by creating several VMs that can be simultaneously running on computation servers (i.e., PMs) at a MEC data center. In this context, by using the NFV platform, physical resources hosted by the PMs at the MEC data center further form a virtualized resource pool, and the virtualized resources are rented out to UEs in the form of VM instances to execute the offloaded computation tasks.

From the MEC data center perspective, the maintenance costs primarily arise from the energy consumption of physical servers, especially with the increasing number of PMs. To reduce the costs, multiple VM instances need to be concurrently placed on top of a single PM catering to the workload requirements of UEs. Due to a large number of PMs hosted by the MEC data center, the mapping of plenty of VM instances to several PMs helps increase the utilization of physical resources and optimize the system performance. Thereby, it is crucial to tackle how to map multiple VM instances to several PMs according to the optimization objectives. One important challenge lies in how to efficiently achieve the VM placement on top of the PMs and allocate the virtualized resources to the UEs satisfying the requirement of workloads.

Many works have been dedicated to the VM placement problem in data centers, and most of them focused on the cloud environments [5]–[7]. In [5], Zhao *et al.* proposed a power-aware and performance-guaranteed placement scheme by formulating the placement problem as a bi-objective optimization model. In [6], Gaggero and Caviglione designed a dynamic model to capture the time evolution law of VM instances running on PMs, and presented a placement mechanism via the model predictive control method. In [7], Liu *et al.* exploited the evolutionary computation policy to analyze the VM placement by minimizing the number of active PMs. Furthermore, some other studies on the VM placement problem for the MEC scenarios have also been reported recently [8], [9]. In [8], Zhao and Liu explored the placement of VM replica copies for supporting resource-intensive and time-critical applications by minimizing the average response time among MEC servers. The latency aware heuristic placement algorithm was presented with lower computation complexity. Based on the mobility prediction of UEs, Plachy *et al.* in [9] proposed a dynamic placement algorithm by defining a reward function from the Markov decision process for communication path selection.

Although the aforementioned studies have made impressive progress on the VM placement problem for not only the cloud environments [5]–[7] but also the MEC scenarios [8], [9], our work differs from them at two points. Firstly, the VM placement problem in their works was formulated bearing in mind an obvious technique constraint that a single VM instance must be running on one PM. This is due to that the existing VM monitor software known as hypervisor fails to support the creation of a single VM instance that spans multiple PMs. However, the vSMP hypervisor from ScaleMP can aggregate multiple PMs into a single highly capable VM, which means that one VM can be technically placed across multiple PMs. Motivated by this observation, our work intends to achieve an energy efficient VM placement by characterizing much more complex mapping relation between VM instances and PMs, which extends the traditional placement constraint. Secondly, the existing works associated with the VM placement over the MEC scenarios primarily focused on the algorithm designs to find an optimal placement result. Nevertheless, our study incorporates the energy consumption during local computing and computation offloading at UEs, apart from the energy consumed by computing at VM instances. Major contributions of this paper can be summarized as:

- We formulate the energy consumption minimization problem as an intractable 0-1 integer linear programming problem with an uncertain number of PMs and unknown linear summation constraints.
- By defining the hyperedge weight as the negative of the sum of energy consumption for computing at VM instances hosted by one PM, we transform the optimization problem into a non-uniform weighted hypergraph model. The energy efficient VM placement is converted to find a maximum-weight hypergraph matching. We propose a hypergraph matching algorithm via local search to seek a maximum-weight subset of vertex-disjoint hyperedges.

II. System Model and Problem Formulation

A. Scenario Description

Consider a MEC system consisting of one eNB integrated with a MEC data center and L mobile UEs denoted by a set $\mathcal{U} \triangleq \{u_1, u_2, \cdots, u_L\}$. Each UE requires plenty of physical resources, e.g., computing resources and memory/storage resources, to match its realistic workload with resource-hungry applications. Particularly, every UE u_k has a total computation workload w_k to be executed within a time duration T, for $k = 1, 2, \cdots, L$. We adopt a quasi-static scenario where the locations of L mobile UEs remains unchanged within the considered time duration. The MEC data center has M PMs denoted by a set $\mathcal{P} \triangleq \{p_1, p_2, \cdots, p_M\}$ to provide the physical resources. For simplicity, we consider two representative resources for a PM, i.e., processor cores and memory sizes. More precisely, we use an available resource vector $\boldsymbol{\Upsilon}_j \triangleq (\alpha_j, \beta_j)$ to characterize PM p_j, where α_j and β_j are the number of available processor cores and the amount of available memory sizes in GB, respectively, for $j = 1, 2, \cdots, M$. By using those available physical resources, the MEC data center can execute the offloaded computation workloads from L UEs. To complete the offloaded task at the MEC data center, UE u_k requires necessary physical resources represented by a resource requirement vector $\boldsymbol{Q}_k \triangleq (q_k^{\mathrm{c}}, q_k^{\mathrm{m}})$, where q_k^{c} and q_k^{m} denote the number of required processor cores and the amount of required memory sizes in GB, respectively.

Let us assume that there are N VM instances denoted by a set $\mathcal{V} \triangleq \{v_1, v_2, \cdots, v_N\}$ that are running on M PMs at the MEC data center. Every VM instance v_i is described by a resource shape vector $\boldsymbol{\Gamma}_i \triangleq (c_i, m_i)$, where c_i and m_i are the number of processor cores and the amount of memory sizes in GB, respectively, for $i = 1, 2, \cdots, N$. It will be reasonable for a UE to purchase a desired number of VM instances to meet the need of its workload due to resource-hungry applications. So we can suppose that each UE can purchase at most δ VM instances from the MEC data center, for $1 < \delta \ll N$.

We use α_{ij} and β_{ij} to denote the number of processor cores and the amount of memory sizes of VM instance v_i that are placed on PM p_j, respectively. Let us further define a binary variable b_{ij} to indicate whether VM instance v_i is placed on PM p_j ($b_{ij} = 1$) or not ($b_{ij} = 0$). Thereby, elements α_j and β_j in $\boldsymbol{\Upsilon}_j$ for PM p_j can be expressed by $\alpha_j = \sum_{i=1}^{N} b_{ij}\alpha_{ij}$ and $\beta_j = \sum_{i=1}^{N} b_{ij}\beta_{ij}$, respectively. Meanwhile, elements c_i and m_i in $\boldsymbol{\Gamma}_i$ for VM instance v_i should at least satisfy the constraints $c_i \geq \sum_{j=1}^{M} b_{ij}\alpha_{ij}$ and $m_i \geq \sum_{j=1}^{M} b_{ij}\beta_{ij}$, respectively. Considering the limited physical resources, let us assume that PM p_j can host at most δ VM instances. Thus, $\sum_{i=1}^{N} b_{ij} \leq \delta$. Additionally, to improve the diversity of virtualized resource usage, it will be also assumed that the virtualized resources of VM instance v_i can be placed across at most d PMs, for $1 \leq d \ll M$. So we have $\sum_{j=1}^{M} b_{ij} \leq d$.

B. Computation Model

We adopt a partial computation offloading model for UEs. That is, the total computation workloads for UEs can either be executed locally at UEs, or be offloaded to and executed by the MEC data center. Here, we use the size of computation data including the program codes and input parameters to describe the total computation workload for each UE. Thereby, for UE u_k, its total computation workload can be expressed as a two-tuple $w_k = \left\{B_k^{\mathrm{local}}, B_k^{\mathrm{off}}\right\}$ bits during time duration T, where B_k^{local} is the data size of computation task executed locally by UE u_k, and B_k^{off} is the data size of computation task computed

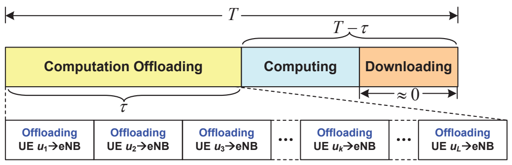

Fig. 1. The time duration allocation for computation offloading using TDMA.

by VM instances.

1) Local Computing: For local computing, the computation task B_k^{local} for UE u_k is executed locally within time duration T. We assume that each UE owns a given number of processor cores as its local computing resources, and every processor core has fixed and equal computing capability. Let ψ_k and f $^{\text{cycles}}/_{\text{s}}$ denote the number of processor cores owned by UE u_k, and the local computing capability of one processor core, respectively. Within time duration T, the number of processor cycles required by UE u_k is given by $Tf\psi_k$. We denote the energy consumption per processor cycle for local computing at UE u_k as ε_k. As a result, the energy consumption for local computing at UE u_k within time duration T is calculated as:

$$E_k^{\text{local}} = Tf\psi_k\varepsilon_k. \tag{1}$$

2) Computation Offloading: Here, the considered time duration T to execute the total computation workload is divided into three stages, i.e., offloading stage, computing stage, and downloading stage, as shown in Fig. 1. For the offloading stage, it will be assumed that the computation task offloading for L UEs is undertaken on the same frequency band with channel bandwidth W_c, and the duration of the offloading stage is τ, for $0 < \tau < T$. To avoid the co-channel multiple access interference during computation offloading within time duration T, we use a TDMA-based media access scheme to schedule multi-UE access to the eNB via uplink transmission link. In the offloading stage, the UEs offload their computation task to the MEC data center one by one during their time slots with equal length τ/L. Let us employ P_k to represent the maximum transmit power of UE u_k during the offloading stage. For convenience, we utilize the quasi-static block fading channel model for multi-UE access to the eNB. Therefore, the maximum transmit power P_k of UE u_k during the offloading stage can be approximately expressed by:

$$P_k(\text{mW}) = 10^{0.1P_0(\text{dBm}) - \varsigma \lg \frac{d_k}{d_0}}, \tag{2}$$

where P_0 is the receiving reference power by the eNB at a reference distance d_0, ς is the path loss exponent, and d_k is the Euclidean distance between UE u_k and the eNB. To simplify analysis, we suppose that each UE has a fixed transmit power which is assumed to be proportional to its maximum transmit power during the offloading stage. More precisely, the transmit power of UE u_k during the offloading stage can be expressed as $p_k = \wp_k P_k$, where $\wp_k \in (0,1]$ is an adjustment factor. Consequently, from the perspective of uplink transmission energy at the UE side, the energy consumption for computation offloading at UE u_k can be written by:

$$E_k^{\text{off}} = \frac{\tau}{L}\wp_k P_k. \tag{3}$$

Based on the Shannon formula, the achievable data rate of UE u_k during the offloading stage can be formulated as $R_k = \log_2\left(1 + \chi \cdot \frac{\wp_k P_k h_k}{\sigma^2}\right)$ $^{\text{bps}}/_{\text{Hz}}$, where σ^2 is the noise power spectral density at the eNB, h_k is the channel gain from UE u_k to the eNB, and χ is the constant processing gain factor depending upon an acceptable bit error rate of the transmission link and the modulation/coding schemes. By taking the communication overhead (e.g., encryption and packer header) denoted by η_k into account, the actual data size of computation task for UE u_k to be offloaded is equal to $\eta_k B_k^{\text{off}}$, which can be further calculated as $\eta_k B_k^{\text{off}} = \frac{W_c\tau}{L}\log_2\left(1 + \chi \cdot \frac{\wp_k P_k h_k}{\sigma^2}\right)$. It is evident that $q_k^{\text{m}} \geq 6.25 \times 10^{-32}\eta_k B_k^{\text{off}}$ for UE u_k to comply with the requirement of the amount of memory sizes.

3) Computing at the MEC data center: For the computing stage, once the actual data size of computation task is offloaded to the MEC data center, the VM instances that are purchased by UEs will execute the received actual computation task. Due to the fact that the eNB has much powerful communication capability than that of UEs, the time to download computation results from eNB to UEs can be neglected. We turn our attention to the energy consumption for computing at each VM instance during the computing stage with duration $T - \tau$. Let us use ϑ (in processor cycles per second) to stand for the computing capability of one processor core hosted by the PM. Considering the constraint of the number of processor cores, the computing capability of VM instance v_i can be easily expressed by ϑc_i. Thus, the number of processor cycles needed by VM instance v_i during the computing stage can be expressed as $(T-\tau)\,\vartheta c_i$. We also denote the energy consumption per processor cycle for computing at VM instance v_i as ξ_i. Therefore, the energy consumption for computing at VM instance v_i during the computing stage can be given by:

$$\begin{aligned} E_i^{\text{vm}} &= (T-\tau)\,\vartheta c_i \xi_i, \\ &= \sum_{j=1}^{M} (T-\tau)\,\vartheta b_{ij}\alpha_{ij}\xi_i. \end{aligned} \tag{4}$$

By incorporating local computing and computation offloading at UE along with computing at VM instance, the total energy consumption of the MEC system for computing and offloading during time duration T can be characterized by:

$$\begin{aligned} E^{\text{total}} &= \sum_{k=1}^{L} E_k^{\text{local}} + \sum_{k=1}^{L} E_k^{\text{off}} + \sum_{i=1}^{N} E_i^{\text{vm}}, \\ &= \sum_{k=1}^{L}\left(Tf\psi_k\varepsilon_k + \frac{\tau\wp_k P_k}{L}\right) \\ &\quad + \sum_{i=1}^{N}\sum_{j=1}^{M} (T-\tau)\,\vartheta b_{ij}\alpha_{ij}\xi_i. \end{aligned} \tag{5}$$

C. Problem Formulation

Under the above setup, our objective is to minimize the total energy consumption for computing and offloading, aiming to obtain the optimal VM placement. Owing to the fixed local computing capability and the maximum transmit power for each UE, the total energy consumption of the MEC system actually depends on the energy consumption for computing at the MEC data center. Therefore, our target is transformed to minimize the energy consumption for computing at the MEC data center. Let $\mathbf{B}_{N\times M} = (b_{ij})_{N\times M}$ represent an $N \times M$ 0-1 VM placement matrix. Then the energy consumption minimization problem can be mathematically formulated as:

$$\underset{M,\mathbf{B}_{N\times M}}{\text{minimize}} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} (T-\tau)\,\vartheta b_{ij}\alpha_{ij}\xi_i \tag{6a}$$

$$\text{subject to} \quad \sum_{j=1}^{M} b_{ij} \le d,\ \forall i, \tag{6b}$$

$$\sum_{i=1}^{N} b_{ij} \le \delta,\ \forall j, \tag{6c}$$

$$\sum_{j=1}^{M} b_{ij}\alpha_{ij} \le c_i,\ \forall i, \tag{6d}$$

$$\sum_{j=1}^{M} b_{ij}\beta_{ij} \le m_i,\ \forall i, \tag{6e}$$

$$b_{ij} \in \{0,1\},\ \forall i,\ \forall j. \tag{6f}$$

From (6), we can observe that the formulated problem is not a simple 0-1 integer linear programming problem in that there exists an integer variable M. Note that some linear summation constraints in (6) are also unknown. When the numbers of PMs and VM instances become large, the problem will be very difficult to solve using traditional exact algorithms. Due to the complex placement relation, we intend to transform our optimization problem into a weighted hypergraph by defining the hyperedge weight as the sum of energy consumption for computing at every VM instance hosted by a PM. In this way, the energy efficient VM placement is achieved through the solution of the weighted hypergraph matching problem. To our best knowledge, the local search has been used as an approximation algorithm to solve such kind of maximum-weight hypergraph matching problem [10], [11]. Having this in mind, in Section III, we will convert the problem into a weighted hypergraph and further propose an effective hypergraph matching algorithm via local search.

III. Virtual Resource Allocation Using Hypergraph Matching Approach

A. Hypergraph Construction

Let us first construct the weighted hypergraph model based on the mapping relation between VM instances and PMs. Every hyperedge is associated with a definite weight, weighting the energy consumption for computing at VM instances. The reason for adopting the hypergraph model rather than the conventional graph theoretic approach is based on two points: i) the placement of every VM instance over PMs is complicated, because every PM hosts a certain number of VM instances and every VM instance is also placed on several PMs; ii) the simple graph model is traditionally used to model pairwise relation between objects, which cannot be used to analyze multi-dimensional mapping relation among objects. Fortunately, hypergraph model wherein its hyperedge can be regarded as a subset of the vertex set is very suited to represent complex mapping relation among multiple objects.

Algorithm 1 Generate An Initial Independent Set $\mathcal{S}$ Based on Sequential Algorithm

Input: Original hypergraph $\mathscr{H}$ with the weighted hyperedges
Output: Initial independent set $\mathcal{S}$ of conflict graph $\mathscr{G}$

```
1: Initialization: $T$, $\mathbf{B}_{N\times M}$, $\alpha_{ij}$, $\beta_{ij}$, $\tau$, $\vartheta$, and $\xi_i$.
2: Generate conflict graph $\mathscr{G}$ with vertex set $\mathcal{Z}$ based on original hypergraph $\mathscr{H}$.
3: Set $\mathcal{S} = \emptyset$.
4: while $\mathcal{Z} \neq \emptyset$ do
5:     Set $\mathcal{A}_{z^{\max}} = \emptyset$.
6:     Select a vertex $z^{\max}$ with the maximum weight of vertex from $\mathcal{Z}$ of conflict graph $\mathscr{G}$.
7:     Find all the adjacent vertices of $z^{\max}$, and let $\mathcal{A}_{z^{\max}}$ be the set of the adjacent vertices of $z^{\max}$.
8:     Set $\mathcal{S} = \mathcal{S} \cup \{z^{\max}\}$ and $\mathcal{Z} = \mathcal{Z} - \{z^{\max}\} \cup \mathcal{A}_{z^{\max}}$.
9: end while
```

Definition 1 *(*Hypergraph*)*: A hypergraph $\mathscr{H}$ is defined as an ordered pair $\mathscr{H} \triangleq (\mathcal{V}(\mathcal{H}), \mathcal{E}(\mathcal{H}))$, where $\mathcal{V}(\mathcal{H})$ is a non-empty finite set of vertices, and $\mathcal{E}(\mathcal{H}) \triangleq \{e_1, e_2, \cdots, e_{\Phi}\}$ is a collection of non-empty subsets of $\mathcal{V}(\mathcal{H})$ called the set of hyperedges, such that $e_\ell \neq \emptyset$ and $\bigcup_{\ell=1}^{\Phi} e_\ell = \mathcal{V}(\mathcal{H})$, for $\ell = 1, 2, \cdots, \Phi$. The element $e_\ell \in \mathcal{E}(\mathcal{H})$ is said to be the hyperedge of hypergraph $\mathscr{H}$. In a weighted hypergraph, each hyperedge e_ℓ is associated with a weight defined by $\omega(e_\ell)$.

For our considered scenario, a VM instance corresponds to a vertex of $\mathscr{H}$, and the set of VM instances is equal to the set of vertices of $\mathscr{H}$, i.e., $\mathcal{V}(\mathcal{H}) = \mathcal{V}$. Owing to the fact that a PM can host multiple VM instances, a PM can refer to a hyperedge of $\mathscr{H}$, and every hyperedge of $\mathscr{H}$ contains at most δ vertices, as depicted in Fig. 2. Thus, we have $\mathcal{E}(\mathcal{H}) = \mathcal{P}$ and $\Phi = M$. Without losing generality, we define the weight of every hyperedge e_ℓ as the negative of the accumulated energy consumption for computing of $\sum_{i=1}^{N} b_{i\ell}$ VM instances hosted by hyperedge e_ℓ, i.e., $\omega(e_\ell) = -\sum_{i=1}^{N}(T-\tau)\,\vartheta b_{i\ell}\alpha_{i\ell}\xi_i$, for $\ell = 1, 2, \cdots, M$. Therefore, our optimization problem can be formally converted into a weighted hypergraph model.

Note that such a weighted hypergraph is a non-uniform weighted hypergraph bounded by the maximum hyperedge size δ. In this context, the existence of a perfect matching with a collection of vertex-disjoint hyperedges that covers all the vertices of $\mathscr{H}$ cannot be completely guaranteed. Alternatively, our target for the energy consumption minimization problem is to find an (M^*)-perfect matching as a collection of M^* vertex-disjoint hyperedges with the maximum total weight by covering as many vertices as possible in $\mathscr{H}$. In other words, we try to obtain the optimal VM placement matrix by seeking a maximum-weight subset of vertex-disjoint hyperedges.

B. Hypergraph Matching Algorithm Design

For a generalized hypergraph matching problem, seeking a maximum-weight subset of vertex-disjoint hyperedges is NP-hard. Inspired by the local search idea with an increasing approximation ratio by $\frac{\gamma}{\gamma-1}$ factor (integer $\gamma \ge 2$) in polynomial time [12], we next use local search to design the hypergraph matching algorithm to find a suboptimal solution.

For ease of discussion, we define a conflict graph $\mathscr{G}$ of original hypergraph $\mathscr{H}$ as an ordinary graph wherein every

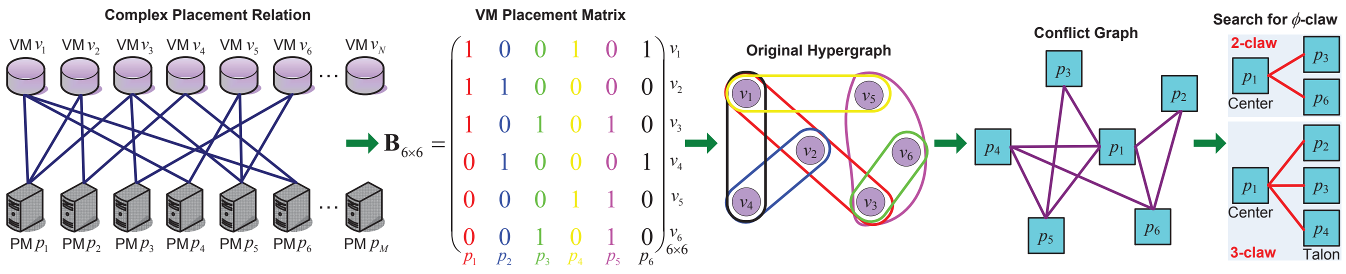

Fig. 2. Illustration of hypergraph construction, conflict graph generation, and ϕ-claw search.

vertex $z_\ell \in \mathcal{Z}$ of $\mathscr{G}$ corresponds to every hyperedge e_ℓ of $\mathscr{H}$, and every edge of $\mathscr{G}$ refers to the adjacent hyperedges if they intersect with each other at least one vertex of $\mathscr{H}$. Note that the weight of every vertex z_ℓ of $\mathscr{G}$ is represented by the weight of the associated hyperedge e_ℓ of $\mathscr{H}$. From Fig. 2, vertex p_1 of the conflict graph stands for hyperedge $\{v_1, v_2, v_3\}$ of the original hypergraph, and the edge between vertex p_1 and vertex p_5 of the conflict graph shows that hyperedges $\{v_1, v_2, v_3\}$ and $\{v_3, v_5, v_6\}$ intersect more than one vertex in the original hypergraph.

Definition 2 *(ϕ-claw $\mathscr{C}_\phi$)*: Based on a conflict graph $\mathscr{G}$, a ϕ-claw is an induced subgraph $\mathscr{C}_\phi$ which consists of an independent set $\mathcal{I}_{\mathscr{C}_\phi}$ of ϕ vertices, called talons, and a center vertex that connects to ϕ independent vertices.

For a given center vertex p_1 of the conflict graph in Fig. 2, we can find 1-claw, 2-claw, and 3-claw. However, we cannot search for a 4-claw for any center vertex of the conflict graph, which means that such a conflict graph is 4-claw free. Let $\mathcal{S}$ and $\Xi(\mathcal{S})$ be the initial independent set with $|\mathcal{S}|$ vertices of conflict graph $\mathscr{G}$ and the cumulative sum of the weight of $|\mathcal{S}|$ vertices in $\mathcal{S}$, respectively. To obtain an initial independent set $\mathcal{S}$ from conflict graph $\mathscr{G}$ with vertex set $\mathcal{Z}$, we utilize a sequential algorithm to approximately obtain a maximum-weight subset of edge-disjoint vertices. The procedure of the adopted sequential algorithm is presented in Algorithm 1.

We use $\mathcal{A}\left(\mathcal{I}_{\mathscr{C}_\phi}, \mathcal{S}\right)$ to denote the set of the adjacent vertices of an independent set $\mathcal{I}_{\mathscr{C}_\phi}$ in the initial independent set $\mathcal{S}$. Based on the local search and the paradigm of ϕ-claw $\mathscr{C}_\phi$, we propose an iterative algorithm to find an (M^*)-perfect matching for the energy consumption minimization problem, which is summarized in Algorithm 2. Based on the output of Algorithm 2, we can determine the optimized variable M^* is equivalent to the cardinality of the maximum-weight subset $\mathcal{E}^*(\mathcal{H})$, i.e., $M^* = |\mathcal{E}^*(\mathcal{H})|$. We remark that the proposed iterative algorithm via local search achieves the optimal VM placement. Compared with other greedy algorithms, e.g., the sequential algorithm, the local search based iterative algorithm improves the overall searching performance by identifying ϕ-claw $\mathscr{C}_\phi$. With the attained energy efficient VM placement , the MEC data center then allocates the virtualized resources to every UE in the form of multiple VM instances hosted by one PM, to meet the requirement of physical resources.

IV. Simulation Results

In this section, we conduct simulations to verify our theoretical analysis and evaluate the performance of our proposed hypergraph matching algorithm. We consider a MEC system consisting of an eNB with a MEC data center and $L = 5$ UEs randomly distributed around the eNB. In the MEC data center, there are $N = 12$ VM instances that provide different shapes catering to the UE's workload requirement. Moreover, each PM hosts at most $\delta = 7$ VM instances, and a VM instance is placed across at most $d = 25$ PMs. For the setup of the VM instance's processor cores and memory sizes, we take advantage of the standard VM instance shapes provided by the Oracle Cloud Infrastructure. Specifically, we choose a portion

Algorithm 2 (M^*)-Perfect Matching Based on Local Search

```
Input: Original hypergraph $\mathscr{H}$ with the weighted hyperedges
Output: Maximum-weight subset $\mathcal{E}^*(\mathcal{H})$ in hypergraph $\mathscr{H}$
 1: Initialization: $T$, $\mathbf{B}_{N\times M}$, $\alpha_{ij}$, $\beta_{ij}$, $\tau$, $\vartheta$, and $\xi_i$.
 2: Generate conflict graph $\mathscr{G}$ based on original hypergraph $\mathscr{H}$.
 3: Obtain an initial independent set $\mathcal{S}$ with $|\mathcal{S}|$ vertices of conflict
    graph $\mathscr{G}$ using Algorithm 1.
 4: Sort $|\mathcal{S}|$ vertices of $\mathcal{S}$ in an ascending order based on the weight
    of every vertex.
 5: Set $l = 1$.
 6: repeat
 7:     Find the set $\Im_l$ of adjacent vertices of $l$-th vertex $\mathcal{S}(l)$ in $\mathcal{S}$,
        and sort $|\Im_l|$ vertices of $\Im_l$ in a descending order based on
        the weight of every vertex.
 8:     Set $l$-th vertex $\mathcal{S}(l)$ as a center vertex of $\phi$-claw $\mathscr{C}_\phi$.
 9:     for $\phi = 2$ to $\delta$ do
10:         Search for $\phi$-claw $\mathscr{C}_\phi$ from the set $\Im_l$ in conflict graph $\mathscr{G}$.
11:         if there exists $\phi$-claw $\mathscr{C}_\phi$ which is further subject ro
            $\left(\Xi\left(\left(\mathcal{S}-\mathcal{A}\left(\mathcal{I}_{\mathscr{C}_\phi},\mathcal{S}\right)\right)\cup\mathcal{I}_{\mathscr{C}_\phi}\right)\right)^2 > (\Xi(\mathcal{S}))^2$ then
12:             $\mathcal{S} = \left(\mathcal{S}-\mathcal{A}\left(\mathcal{I}_{\mathscr{C}_\phi},\mathcal{S}\right)\right)\cup\mathcal{I}_{\mathscr{C}_\phi}$.
13:             go to step 4.
14:         end if
15:     end for
16:     $l = l + 1$.
17: until $l > |\mathcal{S}|$
18: return subset $\mathcal{E}^*(\mathcal{H}) = \left\{e_1^*, e_2^*, \cdots, e_{|\mathcal{S}|}^*\right\}$.
```

TABLE I
SIMULATION PARAMETERS.

Parameter	Value
Euclidean distance between UE and eNB, d_k	$[25, 10, 32, 58, 40]$m
Reference distance, d_0	50 m
Receiving reference power by eNB, P_0	14 dBm
Channel bandwidth, W_c	5 MHz
Path loss exponent, ς	2
Processing gain factor, χ	0.1962
Noise power spectral density at eNB, σ^2	-12 dBm
Local computing capability, f	1.2×10^8 cycles/s
Number of processor cores for UE, ψ_k	$[4, 6, 5, 8, 4]$
Communication overhead, η_k	2000 bits
Duration of offloading stage, τ	0.03 s
Adjustment factor for transmit power, $\wp_k$	0.92

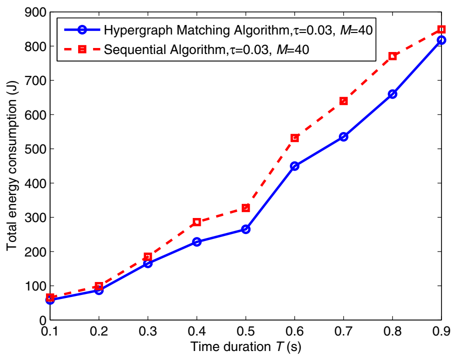

Fig. 3. Comparison of total energy consumption of the MEC system under different time duration with $\vartheta = 10^9$ cycles/s.

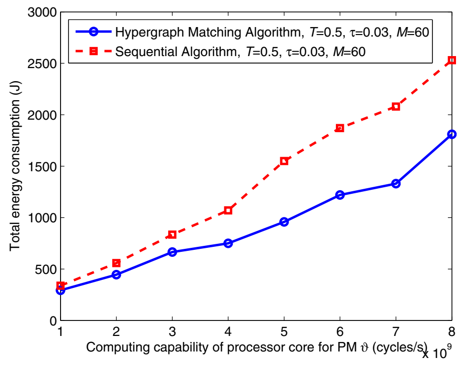

Fig. 4. Comparison of total energy consumption of the MEC system under different computing capabilities of PM's processor core.

of the Oracle's VM instance shapes for every VM instance, i.e., $\alpha_{ij} \in [1, 2, 3, 4, 8, 16]$ and $\beta_{ij} \in [7, 8, 14, 16, 28, 32, 56]$ GB. Furthermore, the energy consumption per processor cycle for local computing at a UE and for computing at a VM instance is set to $\varepsilon_k = 8.7 \times 10^{-8}$ J/cycle and $\xi_i = 1.2 \times 10^{-8}$ J/cycle, respectively. For the benchmark, we consider the sequential algorithm in Algorithm 1 to approximately obtain a maximum-weight subset of edge-disjoint vertices in $\mathscr{G}$. The list of other parameters used in the simulations are summarized in Table I.

In Fig. 3, we examine the performance of the impact of time duration T from 0.1 s to 0.9 s on the total energy consumption of the MEC system with $M = 40$ PMs, offloading duration $\tau = 0.03$ s, and computing capability of one processor core for PM $\vartheta = 10^9$ cycles/s. It can be seen that the total energy consumption of the MEC system increases with the growth of time duration T. This is because the total energy consumption of the MEC system is theoretically proportional to time duration T. Besides, the total energy consumption of the MEC system by using the proposed algorithm is clearly lower than the total energy consumption via the sequential algorithm when time duration T is larger than 0.4 s.

Fig. 4 shows the comparison of total energy consumption of the MEC system between the hypergraph matching algorithm and the sequential algorithm under different computing capabilities of PM's processor core with $M = 60$ PMs, offloading duration $\tau = 0.03$ s, and time duration $T = 0.5$ s. We can find that an increased computing capability of PM's processor core from 10^9 cycles/s to 8×10^9 cycles/s generates the growing energy consumption of the MEC system for both algorithms. Meanwhile, the hypergraph matching algorithm achieves lower energy consumption of the MEC system in comparison with the sequential algorithm under different computing capabilities. The above results further verify the effectiveness of our theoretical analysis and proposed algorithm.

V. Conclusion

In this paper, we studied the energy efficient VM placement and virtual resource allocation problem for the MEC system. The optimization framework of energy consumption minimization for computing and offloading was designed, to obtain the energy efficient VM placement. We converted the optimization problem into a non-uniform weighted hypergraph model, and further developed a local search based hypergraph matching algorithm to find a set of vertex-disjoint hyperedges with the maximum total weight. Thus, the virtualized resources can be assigned to UEs in terms of multiple VM instances. Simulation results show that the proposed algorithm can achieve lower energy consumption of the MEC system under different time durations or computing capabilities of the PM's processor core.

Acknowlegment

This work was partially supported by NSFC (61802107), Hebei Program for Top-notch Young Talents (BJ2017037), and NSF of Hebei (F2019402206, F2017402068, F2018402198).

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th quarter 2017.
[2] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.
[3] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 375–390, Jan. 2019.
[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.
[5] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1385–1400, Jun. 2018.
[6] M. Gaggero and L. Caviglione, "Model predictive control for energy-efficient, quality-aware, and secure virtual machine placement," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 420–432, Jan. 2019.
[7] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.
[8] L. Zhao and J. Liu, "Optimal placement of virtual machines for supporting multiple applications in mobile edge networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, Jul. 2018.
[9] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE PIMRC*, Valencia, Spain, Sep. 2016.
[10] L. Wang, H. Wu, Y. Ding, W. Chen, and H. V. Poor, "Hypergraph-based wireless distributed storage optimization for cellular D2D underlays," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 10, pp. 2650–2666, Oct. 2016.
[11] H. Zhang, L. Song, Y. Li, and G. Y. Li, "Hypergraph theory: Applications in 5G heterogeneous ultra-dense networks," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 70–76, Dec. 2017.
[12] P. Berman, "A d/2 approximation for maximum weight independent set in d-claw free graphs," in *Proc. Algorithm Theory-SWAT 2000*. Springer, Berlin, Heidelberg, Mar. 2002, vol. 1851, pp. 214–219.