Research Article

Joint Virtual Machine Selection and Computation Resource Allocation in Mobile Edge Computing

Huifeng Yang,1 Xianglong Meng,1 Yichao Li,1 Yong Wei,1 Li Shang,1 Jiucheng Wang,1 and Peng Lin2

1Information and Telecommunication Branch, State Grid Hebei Electric Power Co. Ltd., Shijiazhuang 054005, China
2Beijing Vectinfo Technologies Co. Ltd., Beijing 100082, China

Correspondence should be addressed to Peng Lin; linpeng@vectinfo.com

Received 12 May 2023; Revised 21 August 2023; Accepted 1 September 2023; Published 27 September 2023

Academic Editor: Rajkishor Kumar

Copyright © 2023 Huifeng Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Mobile edge computing (MEC) is considered as an effective technology to enhance the storage and computation capability of smart power sensors (SPSs) in smart grid networks. The MEC server is composed of multiple virtual machines (VMs) with powerful computation capability, and each VM can process multiple tasks independently, which cannot be ignored during the task computation period. In this work, we aim to minimize the energy consumption of SPSs subject to the task loading delay by jointly optimizing the VM selection and computation resource allocation. Considering the formulated problem is nonconvex, we first utilize the linearization method to transform it into a convex optimization problem. And then, by using the branch and bound method, we propose the joint VM selection and computation resource allocation (JVMSRA) algorithm. Considering the complexity of the JVMSRA algorithm is high, we decompose the primal problem into two subproblems and solve them by utilizing the ant colony method and CVX package, respectively. Based on the solutions of the two subproblems, the resource allocation-based ant colony (RAAC) algorithm is proposed. Simulation results show that the proposed RAAC algorithm and JVMSRA algorithm decrease by 6% and 8.8% on average compared with the benchmark algorithm, respectively, when the computation resources of each VM increase from 1 to 3GHz.

1. Introduction

The smart grid networks are considered as an optimal solution to solve the reliability, security, and increasing energy demand problems of the traditional power grid networks [1]. In the smart grid networks, numerous smart power sensors (SPSs) are deployed, including temperature sensors, humidity sensors, and current sensors. The data from these sensors generate a substantial number of tasks that require computation [2]. However, due to the limited computation resource and energy of these smart devices, it is difficult to process these tasks. In order to solve these problems, mobile cloud computing (MCC) has been proposed. By offloading these tasks to the powerful cloud center, these tasks can be processed easily. However, the long distance between the SPSs and the cloud center presents a challenge, as it leads to significant task offloading delay [3].

To address the issue of task offloading delay in MCC, a new approach called mobile edge computing (MEC) has been proposed. Because the MEC server is closer to the SPSs, the tasks offloading delay and energy consumption can be reduced easily [4]. Recently, computation offloading to minimize energy consumption has attracted significant attention. Considering the task offloading and result downloading, a joint transmission power, transmission time, and task offloading partition algorithm was proposed to minimize the total energy consumption. The proposed algorithm by utilizing the successive convex approximation method to achieve this goal [5]. To address the constantly changing load conditions on MEC servers and to optimize resource utilization for meeting user quality of service (QoS) requirements, a novel resource allocation algorithm named power migration expand has been proposed with the objective of minimizing the MEC server energy consumption [6]. Xu et al. [7] proposed an algorithm that jointly optimized computation
offloading, data compression, and resource allocation to minimize energy consumption, while considering the latency constraint and finite computation capacity of the MEC system. In practical engineering, the MEC server is composed of multiple virtual machines (VMs) with powerful computation capability, and each VM can process multiple tasks independently. However, the previous works did not consider the architecture of the MEC server and only considered it as a separate server.

This work aims to minimize the energy consumption of SPSs in smart grid networks by jointly optimizing the VM selection and resource allocation. Since the formulated problem is nonconvex, the primal problem is transformed into a convex optimization problem by utilizing the linearization method. Next, we use the branch and bound (BB) method to develop the joint VM selection and resource allocation (JVMSRA) algorithm. Considering the complexity of the JVMSRA algorithm is high, we decompose the primal problem into two subproblems and then solve them by utilizing the ant colony method and CVX package, respectively. Based on the solutions of the two subproblems, the resource allocation-based ant colony (RAAC) algorithm is proposed. Simulation results show that the proposed RAAC algorithm and JVMSRA algorithm decrease by 6% and 8.8% on average compared with the benchmark algorithm, respectively, when the computation resources of each VM increase from 1 to 3GHz. Our main contributions of the paper are summarized as follows.

1. The energy consumption of SPSs minimization problem is formulated, which jointly optimizes the VM selection and resource allocation with considering the task offloading delay and system resources constraints.

2. Considering the coupling of optimization variables and the nonconvexity of the formulated problem, the linearization method is first employed to transform it into a convex optimization problem, and then the JVMSRA algorithm is proposed by using the BB method.

3. Considering the complexity of the JVMSRA algorithm is high, we first decompose the primal problem into two subproblems, and then utilize the ant colony method and CVX package to solve them, respectively. Based on the solutions of the two subproblems, the RAAC algorithm is proposed.

4. Simulation results are provided, which demonstrate the accuracy and effectiveness of both the proposed JVMSRA and RAAC algorithms.

The remainder of this paper is structured as follows. In Section 2, we provide a summary of the related works. Section 3 introduces the system model, which includes the network and computation models. In Section 4, we formulate the SPSs energy consumption minimization problem. Due to the coupling of optimization variables and the nonconvexity of the problem, we use linearization method to transform it into a convex optimization problem. Section 5 presents the JVMSRA algorithm, which utilizes the BB method to solve the primal problem. In Section 6, we propose the RAAC algorithm. Section 7 presents simulation results to validate the effectiveness of the proposed algorithms. Finally, in Section 8, we conclude the paper.

2. Related Works

In order to enhance the MEC system performance and improve resource utilization, the joint task offloading and resource allocation scheme have become a hot topic in recent years. There are mainly three aspects to study the resource allocation.

The first aspect is to maximize the system utility. Considering the influence of time-varying channels in vehicular edge computing scenarios, Li et al. [8] proposed two joint task offloading and resource allocation schemes to maximize the system utility. Considering the dynamically generated tasks exceed the network capacity region, the network will be congested, a stochastic optimization problem was formulated to maximize the system utility and ensure the queue stability. By utilizing the Lyapunov optimization method, a delay-aware task congestion control and resource allocation algorithm was proposed [9]. Xue and An [10] investigated the task offloading problem of multiuser, multitask, and multi-server through joint task offloading and resource allocation to maximize the system utility. Considering the tasks can be offloaded to the nearby microbase station (BS) and macro-BS, by utilizing Lyapunov optimization method, a joint bandwidth and resource allocation algorithm was proposed to maximize the system utility [11]. Considering the dynamic change of slice demands and environment information, a joint radio and computation resource allocation problem was formulated. By utilizing the deep deterministic policy gradient method, the problem has been solved [12]. For the objective of maximizing the system utility in vehicular edge computing networks with both delay-sensitive and nondelay-sensitive tasks, a problem was formulated. To solve this problem, a joint task offloading and resource allocation algorithm was proposed by using the distribution method and the ant colony algorithm [13].

The second aspect is to reduce the delay of task offloading. For the noncollaborative road side unit scenarios and collaborative road side unit scenarios, Li et al. [14] proposed two joint MEC server selection and resource allocation schemes to minimize the total task offloading delay. For the dynamic characteristic of tasks in MEC systems, an online task offloading algorithm was proposed to minimize the task offloading delay while also satisfying the energy consumption constraint [15]. Considering the multiple users can offload the tasks by utilizing the same frequency band, a partial task offloading algorithm was proposed to minimize the task offloading delay [16]. Li et al. [17] formulated a joint task offloading and cache resource allocation problem to minimize the task offloading delay. By utilizing the gated recurrent unit algorithm, a multiagent deep-Q-network was proposed [17]. In order to minimize the average delay of task offloading, a joint bandwidth allocation, user device scheduling, and transmission power allocation problem with considering the
directional millimeter-wave link was formulated. By utilizing the alternative optimization method, the joint bandwidth and transmit power algorithm was proposed [18].

The third aspect is to minimize the system energy consumption or maximize the system energy efficiency. Li et al. [19] proposed a novel approach that utilized energy harvesting technology to minimize the total energy consumption while satisfying the service latency requirement. This was achieved through joint optimization of the task offloading ratio and resource allocation while taking into consideration the downlink transmission time [19]. Lim and Hwang [20] formulated two energy efficiency maximization problems for the spatial division multiple access and time-division multiple access based MEC system. By utilizing Dinkelbach method and difference-of-concave programing method, two joint beamforming design and resource allocation algorithms were proposed [20]. Considering the limited computation resources of the MEC server and the QoS of Internet of things devices in the backscatter-assisted wireless powered MEC networks, a two-layer iterative algorithm was proposed to maximize the system energy efficiency [21]. In order to minimize energy consumption, a general hybrid nonorthogonal multiple access task offloading scheme was proposed, this scheme is low complexity and can achieve Pareto-optimal [22]. Considering the massive connectivity of Internet of things devices, a joint radio and computation resource allocation problem was formulated to maximize the system energy efficiency. By utilizing matching, sequential convex programing algorithm, and the Knapsack method, the primal problem was solved [23]. Considering the BS equipped with renewable energy resources can transfer energy to the smart devices, a total energy minimization problem with task offloading delay constraint was formulated. The authors considered the perfect and imperfect channel state information, two joint computation offloading and resource allocation algorithms were proposed [24]. A task offloading delay minimization problem was formulated for nonorthogonal multiple access assisted MEC. By utilizing the Dinkelbach method and Newton method, two iterative algorithms were proposed [25]. Considering the approximate computing has emerged as a promising method for reducing energy consumption while trading a controllable quality loss, Saraswat et al. [26] introduced an enhanced approximated version of the modified shared invalid cache coherence protocol modified shared invalid approx.

However, the previous works only considered the MEC server as a separate server. In fact, the MEC server is composed of multiple VMs with powerful computation capability, and each VM can process multiple tasks independently. Therefore, the architecture of the MEC server should be considered when the task offloading algorithm is studied.

3. System Model

This section presents the system model of MEC in smart grid networks, which comprises two main components: the network model and the computation model.

3.1. Network Model

In this system, there are I SPSs and one BS which is equipped with a MEC server in the system, as shown in Figure 1. Each SPS has one task to be processed, and we denote the set of SPSs as $\mathcal{I} = \{1, \ldots, I\}$, which can also be viewed as the set of task indices. For the task $i$, we utilize $s_i, c_i$ to express, where $c_i$ is the computation resources to process task $i$, $s_i$ is the data size of task $i$, and $t^{\text{max}}_i$ is the corresponding delay requirement [8]. Each task can be processed by the SPS locally, or by the MEC server. According to Guerrero et al. [27] and Liu et al. [28], the MEC server is composed of a large number of VMs with powerful computation capability. In this paper, the MEC server consists of $K$ VMs, and we define the set of VMs as $\mathcal{K} = \{1, \ldots, K\}$. Considering the tasks can be processed by different VMs, we denote a binary variable as $\alpha_{ik}$ to express that the task $i$ is processed by the VM $k$ with $\alpha_{ik} = 1$, and the task $i$ is processed by the SPS locally with $\sum_{k=1}^{K} \alpha_{ik} = 0$. We define that each task can be only processed by the SPS locally or by one VM. Therefore, we have $\sum_{i=1}^{I} \alpha_{ik} = \{0, 1\}$.

3.2. Computation Model

(1) Local computation model: we define the maximal computation resources of SPS $i$ as $f^\text{loc}_i$, and the computation resources of SPS $i$ to process task $i$ as $f^\text{loc}_{i, f}$. Therefore, the local computation time can be expressed as follows:

$$t^\text{loc}_i = \frac{c_i}{f^\text{loc}_i}. \quad (1)$$

Based on Li et al. [19], the corresponding energy consumption of SPS $i$ is

$$E^\text{loc}_i = \kappa_i (f^\text{loc}_{i, f})^2 c_i, \quad (2)$$

where $\kappa_i$ is the conversion coefficient of SPS $i$, which is depended on the chip architecture of SPS $i$. According to the practical measurement, we set $\kappa_i = 10^{-27}$ in this paper [9, 19]. Considering the maximal computation resources and the task delay requirement, we can obtain the computation resources of SPS $i$ as follows:
\[ f_i^{\text{loc}} = \min \left\{ \frac{c_i}{T_i^{\max}}, f_i^{\text{loc}} \right\}. \quad (3) \]

(2) MEC server computation model: we define the maximal computation resources of VM \( k \) as \( F_k^{\max} \), and define \( f_{ik} \) as the computation resources of VM \( k \) allocated to process task \( i \). Therefore, the computation time for VM \( k \) to process the task \( i \) is

\[ T_{ik}^{\text{VM}} = \frac{c_i}{f_{ik}}. \quad (4) \]

Meanwhile, the energy consumption of VM \( k \) to process the task \( i \) is

\[ E_{ik}^{\text{VM}} = \kappa_k(f_{ik})^2 c_i, \quad (5) \]

where \( \kappa_k \) is the conversion coefficient of VM \( k \), which is depended on the chip architecture of VM. According to the practical measurement, we set \( \kappa_k = 10^{-28} \) in this paper [9, 19].

In this paper, we denote the transmission rate of SPS \( i \) as \( R_i \). Therefore, when the task \( i \) is processed by VM \( k \), the transmission time of task \( i \) is

\[ T_{ik}^{\text{trans}} = \frac{S_i}{R_i}. \quad (6) \]

Therefore, when the task \( i \) is offloaded to the VM \( k \) to process, the processing time can be expressed as follows:

\[ T_{ik} = T_{ik}^{\text{trans}} + T_{ik}^{\text{VM}}. \quad (7) \]

If the task is offloaded to the VM, we denote \( p_{i}^{T} \) as the transmission power of SPS \( i \), and \( p_{i}^{'T} \) as the power consumption of SPS \( i \) in an idle state when the task \( i \) is processed by the VM. Therefore, the total energy consumption of SPS \( i \) can be expressed as follows:

\[ E_{i}^{\text{SPS}} = p_{i}^{T} T_{ik}^{\text{trans}} + p_{i}^{'T} T_{ik}^{\text{VM}}. \quad (8) \]

As the output data size is significantly smaller than the input data size, the time and energy consumption involved in receiving the processing results are neglected [10].

According to the above analysis, the processing time of task \( i \) can be expressed as follows:

\[ T_i = \sum_{k=1}^{K} \alpha_{ik} T_{ik} + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_i^{\text{loc}}. \quad (9) \]

And the energy consumption of SPSs can be expressed as follows:

\[ E_{\text{total}} = \sum_{i=1}^{I} \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) E_{i}^{\text{loc}} + \sum_{i=1}^{I} \left( \sum_{k=1}^{K} \alpha_{ik} E_{i}^{\text{SPS}} \right). \quad (10) \]

### 4. Problem Formulation and Reformulation

In this section, we formulate a problem for jointly optimizing the VM selection and computation resource allocation to minimize the energy consumption of the SPSs. Since this problem is a nonconvex optimization problem, we utilize the linearization method to transform it into a convex problem.

#### 4.1. Problem Formulation

The problem of minimizing SPSs energy consumption can be expressed as follows:

\[
\begin{align*}
\text{(P1)} \quad & \min_{\alpha, f} E_{\text{total}}, \\
\text{s.t.} \quad & T_i \leq T_i^{\max}, \forall i \in \mathcal{I}, \\
& 0 \leq f_{ik} \leq F_k^{\max}, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \\
& \sum_{i=1}^{I} f_{ik} \leq F_k^{\max}, \forall k \in \mathcal{K}, \\
& \sum_{k=1}^{K} \alpha_{ik} = \{0, 1\}, \forall i \in \mathcal{I}, \\
& \alpha_{ik} \in \{0, 1\}, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}. 
\end{align*}
\]

Where \( \alpha = \{\alpha_{ik}, i \in \mathcal{I}, k \in \mathcal{K}\} \) and \( f = \{f_{ik}, i \in \mathcal{I}, k \in \mathcal{K}\} \) are the sets of VM selection, and VM computation resource allocation, respectively. Constraint (11b) is the delay requirement. Constraints (11c) and (11d) limit the computation resources of the selected VMs. Constraints (11e) and (11f) are the VM selection variables constraints.

As the binary variable \( \alpha_{ik} \) and the continuous variable \( f_{ik} \) are coupled, the primal problem (P1) becomes a nonconvex problem which poses challenges for traditional algorithms to solve it effectively. To overcome this issue, we use the linearization method to transform the problem into a convex optimization problem in the following subsection.

#### 4.2. Problem Transformation

To prevent the divide-by-zero, one variables \( \epsilon \) is introduced. We utilize the linearization method to reformulate the problem as follows:

\[
\begin{align*}
\text{(P2)} \quad & \min_{\alpha, f} \sum_{i=1}^{I} \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) E_{i}^{\text{loc}} \\
& + \sum_{i=1}^{I} \left( \sum_{k=1}^{K} \alpha_{ik} \left( p_{i}^{T} \frac{S_i}{R_i} + p_{i}^{'T} \frac{c_i}{f_{ik} + \epsilon} \right) \right), \\
\text{s.t.} \quad & \sum_{k=1}^{K} \alpha_{ik} \left( \frac{S_i}{R_i} + \frac{c_i}{f_{ik} + \epsilon} \right) + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_i^{\text{loc}} \\
& \leq T_i^{\max}, \forall i \in \mathcal{I}. 
\end{align*}
\]
\[(11c) - (11f)\). \quad (12c)\]

**Proposition 1.** The solution of problem (P2) provides a guaranteed lower bound on the solution of primal problem (P1).

**Proof.** See Appendix. \qed

We denote \(\beta_{ik} = 1/f_{ik} + \varepsilon\), the problem (P2) can be reformulated as follows:

\[
\begin{align}
\text{(P3)} \quad & \min_{\alpha, \beta} \sum_{i=1}^{I} \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) E_{i}^{\text{loc}} \\
& + \sum_{i=1}^{I} \left( \sum_{k=1}^{K} \alpha_{ik} \left( p_{i}^{T} \frac{s_{i}}{R_{i}} + p_{i}^{T} c_{i} \beta_{ik} \right) \right), \quad (13a) \\
\text{s.t.} \quad & \sum_{k=1}^{K} \alpha_{ik} \left( \frac{s_{i}}{R_{i}} + c_{i} \beta_{ik} \right) + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_{i}^{\text{loc}} \leq T_{i}^{\text{max}}, \quad \forall i \in \mathcal{J}, \quad (13b) \\
& \frac{1}{\alpha_{ik}} F_{ik}^{\max} + \varepsilon \leq \beta_{ik} \leq \frac{1}{\varepsilon}, \quad \forall i \in \mathcal{I}, \quad \forall k \in \mathcal{K}, \quad (13c) \\
& \sum_{i=1}^{I} \frac{1}{\beta_{ik}} \leq F_{ik}^{\max} + 1 \varepsilon, \quad \forall k \in \mathcal{K}, \quad (13d) \\
& (11e) - (11f). \quad (13e) 
\end{align}
\]

As \(\alpha_{ik}\) is a binary variable, \(\sum_{k=1}^{K} \alpha_{ik}\) and \(\beta_{ik}\) are coupled, the problem remains nonconvex. To solve this problem, we first relax \(\alpha_{ik}\) as \(0 \leq \alpha_{ik} \leq 1\), and then, we introduce \(\omega_{ik} = \alpha_{ik} \beta_{ik}\), considering \(1/\alpha_{ik} F_{ik}^{\max} + \varepsilon \leq \beta_{ik} \leq 1/\varepsilon\) and \(0 \leq \alpha_{ik} \leq 1\), we can get the linearization bound of \(\omega_{ik}\)

\[
\omega_{ik} - \frac{1}{F_{ik}^{\max} + \varepsilon} \alpha_{ik} \geq 0, \quad \forall i \in \mathcal{I}, \quad (14a) \\
\beta_{ik} - \frac{1}{F_{ik}^{\max} + \varepsilon} - \omega_{ik} + \frac{1}{F_{ik}^{\max} + \varepsilon} \alpha_{ik} \geq 0, \quad \forall i \in \mathcal{I}, \quad (14b) \\
\frac{1}{\varepsilon} \alpha_{ik} - \omega_{ik} \geq 0, \quad \forall i \in \mathcal{I}, \quad (14c) \\
\frac{1}{\varepsilon} - \beta_{ik} - \frac{1}{\varepsilon} \alpha_{ik} + \omega_{ik} \geq 0, \quad \forall i \in \mathcal{J}. \quad (14d) 
\]

After substituting \(\omega_{ik}\) into problem (P3), we can get

\[
\begin{align}
\text{(P4)} \quad & \min_{\alpha, \beta, \omega} \sum_{i=1}^{I} \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) E_{i}^{\text{loc}} \\
& + \sum_{i=1}^{I} \left( \sum_{k=1}^{K} \alpha_{ik} p_{i}^{T} \frac{s_{i}}{R_{i}} + \sum_{k=1}^{K} p_{i}^{T} c_{i} \omega_{ik} \right), \quad (15a) \\
\text{s.t.} \quad & \sum_{k=1}^{K} \alpha_{ik} \left( \frac{s_{i}}{R_{i}} + \omega_{ik} \right) + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_{i}^{\text{loc}} \leq T_{i}^{\text{max}}, \quad \forall i \in \mathcal{J}, \quad (15b) \\
& \alpha_{ik} \in [0, 1], \quad \forall i \in \mathcal{J}, \quad \forall k \in \mathcal{K}, \quad (15c) \\
& \sum_{k=1}^{K} \alpha_{ik} \in [0, 1], \quad \forall i \in \mathcal{J}, \quad (15d) \\
& (13c) - (13d). \quad (15e) 
\end{align}
\]

Problem (P4) is a convex optimization problem, which can be efficiently solved by the CVX package. The solution of problem (P4), denoted as \(\bar{E}\), is a lower bound of the primal problem (P1).

We denote \(\mathcal{J}_1 = \{ i | i \in \mathcal{J}, \sum_{k=1}^{K} \alpha_{ik} = 1 \}\) and \(\mathcal{J}_0 = \{ i | i \in \mathcal{J}, \sum_{k=1}^{K} \alpha_{ik} = 0 \}\). If \(\alpha\) is fixed, the primal problem (P1) can be reformulated as follows:

\[
\begin{align}
\text{(P5)} \quad & \min_{f_{i}} \sum_{i \in \mathcal{J}_1} E_{i}^{\text{loc}} + \sum_{i \in \mathcal{J}_0} \sum_{k=1}^{K} \left( p_{i}^{T} \frac{s_{i}}{R_{i}} + p_{i}^{T} \omega_{ik} \right), \quad (16a) \\
\text{s.t.} \quad & \frac{s_{i}}{R_{i}} + \sum_{k=1}^{K} \omega_{ik} \leq T_{i}^{\text{max}}, \quad \forall i \in \mathcal{J}, \quad (16b) \\
& 0 \leq f_{i} \leq F_{i}^{\max}, \quad \forall i \in \mathcal{J}, \quad (16c) \\
& \sum_{i=1}^{I} f_{i} \leq F_{i}^{\max}, \quad \forall k \in \mathcal{K}. \quad (16d) 
\end{align}
\]

Problem (P5) is a convex optimization problem, which can be efficiently solved by the CVX package. The solution of problem (P5), denoted as \(\bar{E}\), is an upper bound of the primal problem (P1).

5. **The Proposed JVMSRA Algorithm**

According to the problem (P4) and problem (P5), we first utilize the BB method to solve the problem (P1), and then the JVMSRA algorithm is proposed.

5.1. The Proposed JVMSRA Algorithm. To employ the BB method by Clausen [29] and Land and Doig [30], we initially create a search tree with the primal problem (P1) as the root node. The optimal solution of problem (P1) is referred to as \(E^\ast\), and we use \(\bar{E}\) to represent its corresponding lower bound.

We also denote the optimal solution of problem (P4) as \(\{ \alpha, \beta, \omega \}\). Based on \(\bar{a}\), we can get \(\bar{\alpha}\) by the following equation.

\[
\bar{\alpha} = \{ \bar{a}_{ik} | \bar{a}_{ik} = \begin{cases} 1 & \alpha_{ik} > 0.5 \\
0 & \alpha_{ik} \leq 0.5 \end{cases} \forall \alpha_{ik} \in \alpha \}. \quad (17) 
\]

When \(\bar{\alpha}\) is fixed, \(\mathcal{J}_1\) and \(\mathcal{J}_0\) are determined, the optimal solution of problem (P5) \(\bar{E}\) can be obtained, and the corresponding resource allocation strategy for the upper bound is
{π, ʃ}. We denote the corresponding resource allocation strategy for the optimal solution of primal problem (P1) as {α∗, f∗}. If E − ʃ ≤ η, where η is the required tolerance, we can obtain {α∗, f∗} = {π, ʃ}. Otherwise, we need to select an unpruned leaf for further branching.

The branching process can be expressed as follows. We select the node l, and from that node we can form the left and right leaf node problems. The left leaf node problem can be expressed as follows:

\[
(P6) \min_{α, f} E_{total}, \quad (18a)
\]

\[
s.t. \quad (11b) - (11f), \quad (18b)
\]

\[
\left\{ \sum_{k=1}^{K} a_{k, l} \alpha_{l} |d(l)|, \sum_{k=1}^{K} a_{k, l} |d(l)| + 1 \right\} = (d(l), 0). \quad (18c)
\]

The right leaf node problem can be expressed as follows:

\[
(P7) \min_{α, f} E_{total}, \quad (19a)
\]

\[
s.t. \quad (11b) - (11f), \quad (19b)
\]

\[
\left\{ \sum_{k=1}^{K} a_{k, l} \alpha_{l} |d(l)|, \sum_{k=1}^{K} a_{k, l} |d(l)| + 1 \right\} = (d(l), 1). \quad (19c)
\]

where d(l) is the determined resource allocation strategy. |d(l)| is the element number of node l. The problems (P6) and (P7) correspond to the left and right leaf nodes, respectively. By utilizing the lower bound obtained from the problem (P4), we can determine the VM selection indicators α∗l and g∗l, respectively, for each leaf node. Then, we can obtain the upper bound of the corresponding VM selection indicators αl, g for every leaf node by solving Equation (17). Finally, by solving the problem (P5), we can derive the upper bound of the values for every leaf node [31, 32].

The bounding process can be expressed as follows. First, the minimal upper bound ʃl and the minimal lower bound ʃl of all unpruned leaf nodes are obtained. If we find the upper bound is smaller than the lower bound, these leaf nodes should be pruned. Second, we can update l = l + 1 until ʃl - ʃl ≤ η. The results are E∗l = ʃl, {α∗, f∗} = {π, ʃ}.

The proposed JVMSRA algorithm is shown in Algorithm 1.

**Algorithm 1:** The proposed JVMSRA algorithm.

1. **Input:** The initial value of π, and ʃ.
2. **Output:** The optimal solution of problem (P9) α∗, and f∗.
3. **Initialize** the system parameters;
4. **Given** l = 1, calculate the upper bound ʃl and the lower bound ʃl according to the problem (P4) and problem (P5);
5. **while** ʃl - ʃl > η do
6. **Select** l = arg minₖ∈l,ʃl Eₖ, and the left and right leaf node problems of node l can be viewed as problem (P6) and problem (P7), respectively;
7. **We calculate** the lower bound values of the left and right leaf nodes, and the corresponding VM selection indicators α²l, and g²l can be determined by solving the problem (P6) and problem (P7), respectively;
8. **By utilizing** (17), we determine the upper bound of the VM selection indicators π′l and π²l for the left and right leaf nodes, respectively;
9. The upper bound values of the left and right leaf nodes are calculated by solving the problem (P5);
10. **We remove** the leaf from the tree;
11. l = l + 1;
12. **end while.**

5.2. The Complexity Analysis. In this subsection, we analyze the complexity of the proposed JVMSRA algorithm. The complexity of problem (P4), (P5), (P6), and (P7) is defined as O(C1), O(C2), O(C3) and O(C4), respectively, as they are all convex problems. Line 2 entails acquiring the initial value, which has a complexity of O(C1 + C2). The while loop in the algorithm has a worst-case complexity of K2 iterations. Lines 4–5 have a complexity of O(C3 + C4 + 2C1), while lines 6–9 have a worst-case complexity of O(1 + 2l). Therefore, the complexity of the proposed JVMSRA algorithm is

\[
O(C1 + C2 + K2(1 + C3 + C4 + 2C1 + 1 + 2l)) = O(K2^2).
\]

6. The Proposed RAAC Algorithm

In the last section, we have proposed the JVMSRA algorithm. However, the complexity of the JVMSRA algorithm is high. Therefore, a low-complexity algorithm should be proposed. In this section, we first transform the primal problem (P1) into two subproblems, and then we utilize the ant colony method and the CVX package to solve them, respectively. Based on the solutions of the two subproblems, the RAAC algorithm is proposed.
6.1. VM Selection Subproblem. The problem (P1) is a non-convex problem, we first fix the computation allocation variable $f$, the VM selection subproblem can be formulated as follows:

$$
\text{(P8)} \min_{\alpha} E_{\text{total}},
$$

\hspace{1cm}

s.t. \quad (11b), (11c), (11f). \tag{21b}

The problem (P8) can be viewed as a matching problem. Based on the ant colony algorithm, we propose a heuristic algorithm to solve this subproblem. We first formulate a bipartite graph $G = (\mathcal{X}, \mathcal{F}, \mathcal{A})$, where $\mathcal{X}$ is the set of VMs, $\mathcal{F}$ is the set of SPSs, and $\mathcal{A} = \{a_{ik} | i = 1, \ldots, I; k = 1, \ldots, K\}$ is the set of edge connecting the SPSs and the VMs. We denote $\lambda_{ik}(t)$ as the connection strength of the pheromone between the SPS $i$ and the VM $k$ at time $t$, which can be expressed as the expectation that SPS $i$ is connected to VM $k$ at time slot $t$. The selection scheme between the SPS and VM can be viewed as the feasible path of the bipartite graph, and the optimal path of the bipartite graph is the optimal solution of the problem [33, 34].

We denote $\eta_{ik}$ as the heuristic factor, which means the cost of SPS $i$ selecting VM $k$. $VM_i^c$ is the remaining computation resources of VM $k$. $p_{ik}$ is the probability of SPS $i$ selecting VM $k$, which can be expressed as follows:

$$
p_{ik} = \begin{cases} 
\frac{\lambda_{ik}(t)\eta_{ik}}{\sum_{k=1}^{K}\lambda_{ik}(t)\eta_{ik}} & VM_i^c > f_{ik} \\
0 & \text{otherwiese}.
\end{cases}
$$

When the VM is selected, we need to subtract the consumed computation resources from the corresponding VM. The pheromone will continue to evaporate with time passes, and we denote the pheromone volatilization coefficient as $\rho$. After one SPS has completed one travel, we select the SPS which obtains the optimal solution to release $\eta_{ik}$ unit of pheromone on its path, the amount of pheromone at time $t+1$ can be expressed as follows:

$$
\lambda_{ik}(t+1) = (1-\rho)\lambda_{ik}(t) + \sum_{k=1}^{K}\eta_{ik}.
$$

When all the SPSs have traversed all the VMs, $K$ solutions can be built. We can substitute the solution to the problem (P8), and obtain the optimal solution. Meanwhile, the optimal path can be obtained.

6.2. Computation Resource Allocation Subproblem. In the last subsection, we obtain the solution of VM selection. In this subsection, the VM selection variable $\alpha$ is fixed, and the computation resource allocation subproblem can be formulated as follows:

$$
\text{(P9)} \min_f E_{\text{total}}, \tag{24a}
$$

s.t. \quad (11b), (11c), (11d).

Algorithm 2: The proposed RAAC algorithm.

1: Input: The initial value of $f$ and $\alpha$.
2: Output: The feasible solution of problem (P9) $\alpha$ and $f$.
3: Initialize the system parameters, such as $K$, $I$, $p_i^a$, $p_i^c$, $s_i$, $T_i$, $\kappa$, $\eta$, and the iterative number $l$.
4: repeat
5: \hspace{0.5cm} Solve the problem (P8) with fixed $f$ and calculate $\alpha$;
6: \hspace{0.5cm} Solve the problem (P9) with fixed $\alpha$ and calculate $f$;
7: Update $l = l + 1$;
8: until Algorithm stopping criterion and convergence.

The problem (P9) is a convex problem, and a lot of standard packages can solve it, such as the CVX package [35].

6.3. The Proposed RAAC Algorithm. By solving the problems (P8) and (P9), we propose the RAAC algorithm to solve the primal problem, as shown in the Algorithm 2. First, given the fixed $f$ and $\alpha$ is obtained by solving the problem (P8). Second, given the fixed $\alpha$ and $f$ is obtained by solving the problem (P9). At last, we can update $l = l + 1$ until the algorithm stopping criterion and convergence.

6.4. The Complexity Analysis. In this subsection, we analyze the complexity of the proposed RAAC algorithm. For the problem (P8), because we utilize the ant colony algorithm, the complexity of problem (P8) is $O((IK)^3)$. For the problem (P9), if we utilize the interior point method, the complexity of problem (P9) is $O(I^3K^2)$. Therefore, the complexity of the proposed RAAC algorithm is $O(I^3K^1.5)$.

7. Simulation Results and Discussion

In this section, we give the simulation results of the proposed JVMSRA algorithm and RAAC algorithm. The experiments are implemented in MATLAB R2018a on a desktop computer with an Intel Core i7-8750H 2.2GHz CPU and 16GB RAM. There are one BS in the cell center, and 10 SPSs are deployed randomly. The transmission power of each SPS is 0.1W, the power consumption of each SPS in idle state is 0.01W, the computation capability of each SPS is 500MHz, and the conversion coefficient of each SPS is $10^{-27}$. The transmission rate is 1,000KB/s. The BS is equipped with one MEC server, and the MEC server is composed of five VMs. The computation capability of each VM is 2GHz, and the conversion coefficient of each VM is $10^{-28}$. The size of the files and the required computation resources for each task follow Gaussian distributions, with $s_i \sim \mathcal{N}(500, 100)$ and $c_i \sim \mathcal{N}(1,000, 100)$, respectively. The file size is measured in KB and the computation resources are measured in megacycles. Each task has a delay requirement of 2.5 s.

The proposed VMSRA algorithm and RAAC algorithm are compared with the other three algorithms to verify the effectiveness.
(1) All local computing (ALC) algorithm: all the tasks are computed by the SPSs locally in this algorithm.

(2) Random VM selection (RAMS) algorithm: all the tasks select the VMs randomly in this algorithm.

(3) VM selection based matching (VMSM) algorithm: in this algorithm, the VM selection subproblem is solved by the matching-coalition game method [36]. And the computation resource allocation subproblem is solved by the CVX package.

Figure 2 is the SPSs energy consumption versus computation resources of each VM. The ALC algorithm, because all the tasks are computed by the SPSs locally, the SPSs energy consumption does not change as the computation resources of each VM increase. For the other four algorithms, the SPSs energy consumption decreases with respect to the computation resources of each VM. This is because as the computation resources of each VM increase, the task computation time decreases, resulting in a reduction in SPSs energy consumption. The SPSs energy consumption of the proposed RAAC algorithm and JVMSRA algorithm decreases by 6% and 8.8% on average compared with the VMSM algorithm, respectively, when the computation resources of each VM increase from 1 to 3GHz.

Figure 3 shows the SPSs energy consumption versus the number of VMs under different algorithms. From this figure, we can see that all the algorithms of SPSs energy consumption decrease with respect to the number of VMs except the ALC algorithm. This is because with the increases of the number of VM, more computation resources can be utilized, and the task computation time can be reduced.

Figure 4 illustrates the SPSs energy consumption versus computation resources requirement of each task under the different algorithms. The SPSs energy consumption is higher with higher computation resource requirements. With the increasing of computation resources requirement of each task, more computation resources are needed, and the task computation time increases. For the proposed RAAC algorithm and JVMSRA algorithm, when the computation resource requirement reaches about 1.2GHz, the SPSs energy consumption reduces 2.4% and 5.4%, respectively, compared with the VMSM algorithm.

Figure 5 illustrates the SPSs energy consumption versus the number of SPSs under the different algorithms. As can be
observed, the energy consumption of SPSs increases as the number of SPSs increases. This phenomenon can be explained that as the number of SPSs increases, more tasks need to be computed, resulting in higher energy consumption by the SPSs.

Figure 6 shows the SPSs energy consumption versus the delay requirement of each task under the different algorithms. From this figure, it is evident that the SPSs energy consumption decreases as the delay requirement of each task increases. This can be explained that when the delay requirement of each task is higher, fewer computation resources can be allocated to each task, resulting in lower overall energy consumption by the SPSs.

8. Conclusion

In this paper, we have formulated a JVMSRA problem to minimize the SPSs energy consumption in the smart grid while considering the task offloading delay constraints. Because the VM selection and the computation resource allocation are coupled, the problem is a nonconvex problem. In order to solve this problem, we first utilized the linearization method to transform it into a convex optimization problem, and then the JVMSRA algorithm was proposed by using the BB method. Considering the complexity of the JVMSRA algorithm cannot be guaranteed, the primal problem was decomposed into two subproblems. The ant colony method and CVX package were utilized to solve the two subproblems, respectively. Based on the solutions of the two subproblems, the RAAC algorithm was proposed. Simulation results have shown that the proposed JVMSRA and RAAC algorithms can achieve better performance compared with the other benchmark algorithms. In practical engineering, the proposed two JVMSRA algorithms can be chosen according to the actual situation.

For this paper, there are three aspects can be extended. (i) This paper only considered the MEC server is composed of multiple VMs which can process all types of tasks. In some scenarios, some specific tasks can only be processed by the specified VMs. Therefore, the joint VM selection and resource allocation strategies for specific tasks can be considered in the future works. (ii) This paper only considered the energy consumption of the SPSs, the power consumption of VMs needs to be considered. Based on the first work, we will focus on the joint VM selection and resource allocation to minimize the total power consumption of the system. (iii) The VMs can be in active/inactive model according to the fluctuating tasks, we will consider the active/inactive model of VMs and resource allocation to minimize the power consumption.

Appendix

Proof of Proposition 1. For the primal problem (P1) and problem (P2), the only differences lie in the objective function and constraints (11b) and (12b). Meanwhile, (12b) is stricter than (11b).

\[
\sum_{k=1}^{K} \alpha_{ik} \left( \frac{s_i}{R_i} + \frac{c_i}{f_{ik}} \right) + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_{loc}^i \geq \sum_{k=1}^{K} \alpha_{ik} \left( \frac{s_i}{R_i} + \frac{c_i}{f_{ik} + \epsilon} \right) + \left( 1 - \sum_{k=1}^{K} \alpha_{ik} \right) T_{loc}^i.
\] (A.1)

We also observe that the objective function value of primal problem (P1) is always greater than or equal to the objective function value of problem (P2). Therefore, the solution of problem (P2) provides a lower bound for the optimal solution of primal problem (P1).
Data Availability
The data used to support the findings of this study are available from the corresponding authors upon request.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
This work was partly supported by State Grid Hebei Electric Power Co. Ltd. Science and Technology Project “Research and application of key technologies for wireless converged networking of new power systems” (SGHEXT00GCJS2200023).

References


