IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 31, 2019, accepted November 28, 2019, date of publication December 3, 2019,
date of current version December 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957340

Stochastic Virtual Machine Placement for Cloud Data Centers Under Resource Requirement Variations

JUNLONG ZHOU, (Member, IEEE), YI ZHANG, LULU SUN, SISI ZHUANG, CHENG TANG, AND JIN SUN, (Member, IEEE)
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

Corresponding author: Jin Sun (sunj@njust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61802185 and Grant 61872185, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180470, in part by the Fundamental Research Funds for the Central Universities under Grant 30919011233 and Grant 30919011402, in part by the Jiangsu Planned Projects for Postdoctoral Research Funds under Grant 2019K025, and in part by the Open Research Project of The Hubei Key Laboratory of Intelligent Geo-Information Processing under Grant KLIGIP-2018A04.

ABSTRACT In cloud computing environment, the optimal placement of virtual machines (VMs) onto physical servers has been of great importance to improving the resource utilization and energy efficiency of data centers. In this work, we study the VM placement problem for minimizing the total energy consumption in a data center under the uncertainty of resource requirements demanded by the VMs. Instead of using deterministic values to represent the resource requirements, as in most existing placers, we propose a stochastic placement approach in which the resource requirement variations are modeled as random variables. We further formulate the uncertainty-aware VM placement problem as a stochastic optimization model, of which the optimization objective is to minimize the total energy consumed by all physical machines (PMs). In the presence of varying resource requirements, the optimization model is subject to a probabilistic constraint on resource overflow probability on each PM (i.e., the probability of demanded CPU/memory exceeding the maximum capacity the PM can provide). To solve this stochastic optimization problem, we develop an efficient metaheuristic to seek for an optimized VM placement solution that minimizes the total energy cost while satisfying the probabilistic resource constraint. Moreover, by incorporating a solution initialization procedure and a neighborhood search strategy, we can further improve the effectiveness of the metaheuristic in solution space exploration. Extensive simulations are performed to justify the proposed approach, in terms of both solution feasibility and energy efficiency. By taking into account the uncertainty of resource requirements, the stochastic method can achieve more energy-efficient placement solutions compared with the deterministic VM placement algorithm.

INDEX TERMS Data center, virtual machine placement, energy efficiency, stochastic optimization.

I. INTRODUCTION

Cloud computing has been the popular computing paradigm that enables users to utilize computing resources from cloud data centers in a pay-as-you-go manner [1]–[3]. As a result, more and more users are willing to submit their commercial or scientific applications to cloud computing platforms for execution, in order to avoid the significant cost of building private computing environments [4]. To consolidate physical resources and deliver on-demand services, data centers use virtualization techniques to host multiple virtual machines (VMs) on a single server or physical machine (PM). Each VM can act as a complete system to execute the applications submitted by users independently [5].

VM placement algorithms, which are responsible for the provisioning of physical resources, are of great importance to the resource utilization of data centers. When performing VM placement, it is necessary to guarantee that each PM is capable of providing sufficient resources (CPU and memory etc.) to tackle the workload from the VMs placed on

The associate editor coordinating the review of this manuscript and approving it for publication was Xiping Hu.

it [6]. Energy efficiency is another important concern when determining the VM placement solution. As reported in the literature, in most data centers over 70% of peak energy are consumed by idle computing nodes [7], [8]. In other words, the low utilization of computing resources leads to a significant amount of energy wastage. Therefore, the VM placer is also expected to improve data center's energy efficiency when executing realistic applications.

A considerable amount of research has been done regarding VM placement strategies [9]–[11]. Most existing approaches are based on the assumption that the amount of physical resources required by each VM is deterministic. In other words, they fail to take into account the fluctuation of resource requirements, which in general exists in practical scenarios. When executing realistic applications, VM workload and consequently the amount of resources demanded by the VM can be varying under different dynamic circumstances [12]–[14]. As a result, in the presence of uncertain resource requirements, deterministic VM placement approaches may degrade the quality of service (QoS) of data centers. To be more specific, the over provisioning of physical resources would cause wastage of costly hardware resources, and the under provisioning of physical resources would lead to unsatisfactory application performance due to insufficient resources [15].

To address the above-mentioned limitations in deterministic VM placement algorithms, this paper presents a stochastic approach that takes into account resource requirements variations to determine uncertainty-aware energy-efficient placement solutions.[1] Instead of using a deterministic value to represent the resource demand, as in most existing placers, we model the uncertainty of resource requirements as random variables, and further formulate the uncertainty-affected VM placement problem as a stochastic optimization problem, of which the optimization objective is to minimize the energy consumption of all PMs. Under uncertain resource requirements, the placement problem is subject to a probabilistic constraint on the resource overflow probability (i.e., the probability of demanded CPU/memory exceeding the maximum capacity the PM can provide). To solve the resultant optimization model, we propose a particle swam optimization (PSO)-based metaheuristic algorithm in which a mapping operator is developed to link each particle with a placement solution. We further incorporate a solution initialization procedure and a neighborhood search strategy into the uncertainty-aware approach, for the purpose of exploring more energy-efficient VM placement solutions. Below we summarize the technical contributions of this work:

- We formulate a stochastic VM placement framework that imposes probabilistic constraints on the resource utilizations on individual PMs.
- We develop an estimation method to predict the resource overflow probability on each PM under resource requirement variations.
- We propose an efficient metaheuristic algorithm to seek for an optimized VM placement solution that minimizes the total energy cost while satisfying the probabilistic resource constraint.
  - We design a mapping operator to convert an individual particle in the swarm into a feasible VM placement solution.
  - We present a solution initialization method to provide the metaheuristic algorithm with a good start.
  - We use a neighborhood search strategy to identify better solutions in the neighborhood of the obtained best solution during each iteration.

Extensive simulations are performed to justify the stochastic placement approach, in terms of both solution feasibility and energy efficiency. By taking into account the uncertainty of resource requirements, the proposed method can achieve more energy-efficient placement solutions compared with traditional deterministic VM placement algorithms, with a guaranteed limit of resource overflow probabilities.

The rest of this paper is organized as follows. Section II surveys existing approaches related to this work. Section III formulates the stochastic VM placement problem that involves probabilistic constraints on resource overflow probabilities. Section IV details the method for predicting the overflow probabilities under resource requirement variations. Section V presents the metaheuristic algorithm developed for solving the formulated stochastic problem, followed by simulation results in Section VI. Conclusions are drawn in Section VII finally.

II. RELATED WORK

VM placement approaches for cloud data centers have been intensively studied in recent years. These approaches aim at finding the optimal mapping from VMs onto PMs with different objectives, such as maximizing resource utilization [17]–[19], minimizing energy efficiency [20]–[22], and improving system performance [23], [24]. In what follows, we specifically discuss those placement methods that are closely related to this work.

In the literature, VM placement problems are generally formulated as NP-hard integer programs [25]. Many metaheuristic algorithms are incorporated into existing VM placement schemes for solution space exploration. For instance, Kanagavelu et al. [26] proposes a greedy search-based algorithm for VM placement with improved resource usage and reduced system traffic. Wang et al. [27] presents an energy-efficient placement algorithm by redefining the coding scheme of PSO framework, for the purpose of finding an appropriate placement solution with lowest energy consumption. The VM placement approach in [28] uses a multi-objective genetic algorithm to jointly optimize resource utilization, power efficiency and thermal costs. Gao et al. [29] presented an ant colony optimization (ACO)-based algorithm seeking for a set of non-dominated solutions to improve resource utilization as well as energy efficiency. The server and VMs are mapped to ant colony and food sources, respectively.

[1] The preliminary version of this paper appeared in [16].

The VM placement solution is modeled as a pheromone trail in the ACO framework. During each iteration, all solutions are evaluated, and the best solutions among them will be used in the next iteration.

There have been several attempts to cope with the VM placement problem in the presence of resource requirement variations. Chaisiri et al. [30] presents a two-stage optimization approach for the VM placement problem suffering the uncertainty of on-demand resources. This approach divides the resource provisioning procedure into reservation phase, utilization phase, and on-demand phase. At the first stage, a preliminary placement solution is explored to specify the resource provisioned in reservation phase. At the second stage, the provisioning of on-demand resources is finely tuned for both phases. However, the VM placement solution at each stage is determined by a deterministic optimization procedure. Thus, this approach is not a purely stochastic optimization procedure that incorporates probabilistic constraints into the VM placement framework. Chen et al. [20] proposes a feasibility-driven VM placement method that takes into account uncertain resource requirements. This method starts with generating an uncertainty-unware placement solution, then uses a sampling-based method to evaluate the feasibility of placement solution under random variations of resource requirements. Driven by the evaluated feasibility, the associated parameters in the uncertainty-unaware placement problem will be updated and the placement solution is tuned. This procedure is iterated until the feasibility requirement is satisfied. The main drawback of this method is that it relies on a random sampling procedure to evaluate each placement solution's feasibility, which is time-consuming, especially for large-scale placement problems.

Different from above-mentioned uncertainty-aware placement approaches, the stochastic method proposed is this work explicitly incorporates probabilistic constraints on resource overflow probabilities into the optimization framework. More importantly, our method is not sampling-based, since we use an analytical model to evaluate the overflow probabilities on individuals PMs. An effective metaheuristic algorithm consisting of several key strategies is developed to solve this stochastic VM placement problem.

Our previous work [16] tackles the same VM placement problem under resource requirement variations as in this work, and proposes a metaheuristic algorithm based on the PSO strategy to find a solution to the stochastic VM placement problem. However, the PSO-based algorithm is not globally optimal, and can be easily trapped in local optima during the search procedure. To enhance the effectiveness of the metaheuristic in solution space exploration, in this extended work, we incorporate a solution initialization procedure to obtain a good initial solution for the metaheuristic algorithm, and employs a neighborhood search strategy to further improve the quality of the best solution explored during each iteration. As will be demonstrated in experimental results, compared with our previous work, the quality of the uncertainty-aware placement solution has been considerably improved by incorporating the above-mentioned new strategies.

III. PROBLEM FORMULATION

In this section, we first discuss the traditional deterministic VM placement problem for minimizing the energy consumption of the data center. With the consideration of uncertain resource requirements, we further formulate the optimization model for the stochastic VM placement problem that involves probabilistic resource constraints.

A. DETERMINISTIC VM PLACEMENT

We assume a data center consisting of a cluster of PMs and a set of VMs running applications submitted by cloud users. Each PM can host multiple VMs, and each VM demands a certain amount of computing resources (CPU and memory) provided by the PM [16]. The goal of VM placement is to determine the optimal VM-to-PM mapping relations that leads to the lowest energy consumption. In addition, each PM can afford sufficient CPU and memory resources demanded by all the VMs placed on it.

Let n and k denote the number of VMs to be placed and the number of available PMs, respectively. We further use c_{ij} and m_{ij} to represent the amounts of CPU and memory demands if VM i is placed on PM j. For the sake of simplicity, c_{ij} and m_{ij} are normalized to the maximum CPU and memory resources PM j can provide. In other words, c_{ij} and m_{ij} are percentage values within $[0,\ 1]$.

We use a vector of binary variables x_{ij}'s to represent the VM placement solution, i.e.,

$$X = [x_1, x_2, \cdots, x_n]^T, \tag{1}$$

where each element $x_i \in \{1, 2, \cdots, k\}$ indicates the index of PM that the i-th VM should be placed on. The vector of x_i values represents a feasible placement solution indicating the mapping of all VMs onto PMs. We further introduce a set of auxiliary variables to formulate the VM placement problem for minimizing energy cost. Let A_{ij} denote the mapping relationship between VM i and PM j, i.e.,

$$A_{ij} = \begin{cases} 1, & \text{if } x_i = j. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Let B_j further denote the status of the corresponding PM j. B_j is 0 if no VM is placed on PM j, and on the other hand, B_j is 1 as long as one VM has been placed on this PM. Consequently, B_j can be expressed as

$$B_j = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} A_{ij} = 0. \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Note that A_{ij} and B_j are dependent on decision variables x_i's. For each placement solution to be evaluated, the A_{ij} and B_j values can be uniquely determined.

Following the energy model in [20], [31], the energy consumption of each PM is composed of a static part and a

dynamic part. The static energy is a constant value as long as the PM is turned on. The dynamic energy, however, has a roughly linear dependency upon the PM's CPU utilization. Hence, the energy cost of the j-th PM can be estimated by

$$E_{\text{PM}}^{j} = E_{\text{stat},j} + \lambda \cdot \sum\nolimits_{i} c_{ij}, \tag{4}$$

where $E_{\text{stat},j}$ is the PM's static energy cost, c_{ij} denotes the amount of CPU resource occupied by each VM hosted by this PM, and λ is a constant coefficient implying the linear dependency of dynamic power upon CPU utilization. The total energy of the data center can be calculated by summing over all individual PMs' energy costs, that is

$$E_{\text{total}} = \sum_{j=1}^{k} E_{\text{PM}}^{j}. \tag{5}$$

If a PM hosts at least one VM, its energy cost can be further expressed in terms of the variables defined previously as

$$E_{\text{PM}}^{j} = \sum_{j=1}^{k} (A_{ij} \cdot \lambda \cdot c_{ij}) + B_j \cdot E_{\text{stat},j}. \tag{6}$$

We consider CPU and memory resource limits in the VM placement framework. Each PM's CPU utilization by all VMs placed on it cannot exceed an upper bound, in order to avoid CPU overflow on the PM, i.e.,

$$\sum_{i=1}^{n} (A_{ij} \cdot c_{ij}) \leq C_j, \quad \forall\, 1 \leq j \leq k, \tag{7}$$

where C_j is maximum amount of CPU utilization by all VMs on PM j. Similarly, we have the following memory resource constraint for each PM:

$$\sum_{i=1}^{n} (A_{ij} \cdot m_{ij}) \leq M_j, \quad \forall\, 1 \leq j \leq k, \tag{8}$$

where M_j is the memory resource capacity of PM j. To summarize, the deterministic VM placement is formulated as an integer program (IP) to find the optimal mapping vector X that minimizes the total energy consumption while satisfying the resource constraints for all individual PMs.

B. STOCHASTIC VM PLACEMENT

In the deterministic VM placement model, the CPU and memory requirements are assumed to be constant values. However, when the data center is executing realistic applications, the amount of physical resources required by each VM may have dynamic variations due to the varying workload assigned onto the VM. Failing to take into account resource requirement variations may cause over provisioning or under provisioning of hardware resources [15], [16]. if hardware resources are overly provisioned, an energy overhead would be incurred. More importantly, if hardware resources are conservatively provisioned, CPU or memory overflow may happen on the PMs, which would in turn deteriorate the QoS of the data center in executing cloud user applications.

We use an illustrative example to demonstrate the importance of stochastic VM placement in the presence of varying resource requirements. Assume that there are three VMs to be placed. The CPU and memory requirements for VM1, VM2, and VM3 on PM1 are (40%, 30%), (40%, 30%), and (30%, 30%), respectively. The resource limit is set as 90% for CPU and memory resources. In deterministic placement framework, since PM1 cannot provide sufficient amount of CPU resource, one more PM has to be turned on to host all three VMs. Assume that the CPU requirement of VM1 drops from 40% to 30% at runtime. In this scenario, the CPU resource on PM1 is sufficient to host all three VMs, and the original placement solution cause over provisioning of physical resources. We further assume that all three VMs are placed on PM1 to save energy cost, however, the memory requirement of VM2 increases by 10% at a subsequent time point. This variation in memory requirement would cause memory overflow on PM1, incurring VM migration and considerable operational cost.

In what follows, we present the stochastic version of VM placement framework that takes into account the uncertainty of resource requirements. To be specific, we assume the resource requirement for each VM is a random variable rather than a deterministic value. For a better interpretation, we use $\tilde{c}_{ij}$ and $\tilde{m}_{ij}$ to represent the uncertain CPU and memory requirements. Affected by this uncertainty, the CPU and memory demand by VMs become variational as well. During VM placement procedure, it is possible that the resource requirement may exceed the resource limit due to these variations. To cope with this situation, we replace the deterministic resource constraints in Eqs. (7) and (8) by the corresponding probabilistic constraints [16]. For this reason, we define α and β to denote the CPU and memory overflow probabilities on PM j, respectively, indicating the probabilities that CPU and memory requirements exceed the PM's resource capacities:

$$OF_j^c = \text{Prob}\left\{\sum_{i=1}^{n} (A_{ij} \cdot \tilde{c}_{ij}) > C_j\right\}, \tag{9}$$

$$OF_j^m = \text{Prob}\left\{\sum_{i=1}^{n} (A_{ij} \cdot \tilde{m}_{ij}) > M_j\right\}. \tag{10}$$

The proposed stochastic VM placement framework aims at determining the most appropriate placement solution such that the energy consumed by all PMs is minimized, and for each PM the resource overflow feasibility under resource requirement variations is guaranteed to be lower than a upper bound limit. We summarize below the complete optimization model for this problem:

$$\text{minimize } E_{\text{total}} = \sum_{j=1}^{k} E_{\text{PM}}^{j} \tag{11}$$

$$\text{subject to Prob}\left\{\sum_{i=1}^{n} (A_{ij} \cdot \tilde{c}_{ij}) > C_j\right\} \leq \alpha_{\max} \tag{12}$$

$$\text{Prob}\left\{\sum_{i=1}^{n} (A_{ij} \cdot \tilde{m}_{ij}) > M_j\right\} \leq \beta_{\max} \tag{13}$$

where $\alpha_{\max}$ and $\beta_{\max}$ are two pre-specified threshold values for CPU and memory overflow probabilities on each PM, respectively. It is required that the placement solution not only minimizes the total energy consumption, but also satisfies the probabilistic resource constraints under resources requirement variations. Due to the existence of the probabilistic constraints in Eqs. (12) and (13), the stochastic placement problem studied in this work is a complicated stochastic optimization model.

IV. PREDICTION OF OVERFLOW PROBABILITIES

A essential step in the stochastic VM placement framework is to predict the resource overflow probabilities for a given VM placement solution. Use the constraint for CPU overflow probability for example. Assume that the varying CPU resource requirement obeys a normal distribution $\tilde{c}_{ij} \sim N\left(\mu_{ij}^c, (\sigma_{ij}^c)^2\right)$. Following the probability theory [32], the total CPU utilization of PM j, i.e., $\sum_{i=1}^{n}(A_{ij} \cdot \tilde{c}_{ij})$, is also a random variable subject to a normal distribution, of which the mean value and variance are given by

$$\eta_c^j = \sum_{i=1}^{n}(A_{ij} \cdot \mu_{ij}), \tag{14}$$

$$\left(\gamma_c^j\right)^2 = \sum_{i=1}^{n}\left(A_{ij} \cdot \sigma_{ij}^c\right)^2. \tag{15}$$

We define a new variable $Z_j = \frac{\sum_{i=1}^{n}(A_{ij}\cdot\tilde{c}_{ij}) - \eta_c^j}{\gamma_c^j}$ to transform the normally distributed CPU utilization into a standard normal variable. Accordingly, the CPU overflow probability can be expressed as

$$\begin{aligned}
&\text{Prob}\left\{\sum_{i=1}^{n}(A_{ij} \cdot \tilde{c}_{ij}) > C_j\right\} \\
&= \text{Prob}\left\{\frac{\sum_{i=1}^{n}(A_{ij} \cdot \tilde{c}_{ij}) - \eta_c^j}{\gamma_c^j} > \frac{C_j - \eta_c^j}{\gamma_c^j}\right\} \\
&= 1 - \text{Prob}\left\{Z_j \le \frac{C_j - \eta_c^j}{\gamma_c^j}\right\}.
\end{aligned} \tag{16}$$

Since the transformed variable Z_j in Eq. (16) follows the standard normal distribution, the CPU overflow probability on PM j can be calculated by

$$OF_c^j = \text{Prob}\left\{Z_j > \frac{C_j - \eta_c^j}{\gamma_c^j}\right\} = 1 - \phi\left(\frac{C_j - \eta_c^j}{\gamma_c^j}\right), \tag{17}$$

where $\phi(z)$ is the cumulative distribution function of a standard normal variable, and is given by

$$\phi(z) = \text{Prob}\{Z \le z\} = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}z^2\right\}. \tag{18}$$

Therefore, we can obtain the CPU overflow probability by calculating the cumulative probability of a standard normal variable in Eq. (17). On the other hand, the memory overflow probability on PM j can be determined in a similar manner, and is given by

$$OF_m^j = 1 - \phi\left(\frac{M_j - \eta_m^j}{\gamma_m^j}\right), \tag{19}$$

where η_m^c and η_m^j are the mean value and variance of the PM's total memory utilization $\sum_{i=1}^{n}(A_{ij} \cdot \tilde{m}_{ij})$, respectively. Algorithm 1 presents the pseudocodes for predicting overflow probabilities given a VM placement solution. Only if the CPU and memory overflow probabilities on each PM are within their threshold values, this solution is considered a feasible solution to the stochastic VM placement problem.

Algorithm 1 Prediction of Overflow Probabilities

Input: i) a placement solution X;
ii) overflow probability limits $\alpha_{\max}$ and $\beta_{\max}$;
Output: CPU overflow probabilities OF_c and memory overflow probabilities OF_m on all PMs;

```
initialize OF_c as a zero vector;
initialize OF_m as a zero vector;
determine A_ij values according to X;
determine B_j values according to A_ij values;
for j = 1 to k do
    set U_C ← 0 ;        /* CPU utilization */
    set U_m ← 0 ;    /* memory utilization */
    for i = 1 to n do
        U_c = U_c + A_ij · c_ij;
        U_m = U_m + A_ij · m_ij;
    end
    calculate CPU overflow probability Prob{U_c > C_j}
    according to Eq. (17);
    update OF_c by setting OF_c^j ← Prob{U_c > C_j};
    calculate memory overflow probability according to
    Eq. (19) and update OF_m;
end
return OF_c and OF_m;
```

V. THE PROPOSED VM PLACEMENT ALGORITHM

This section details the metaheuristic algorithm developed for solving the stochastic VM placement problem formulated in Eqs. (11)-(13). Figure 1 summarizes the general flow of this algorithm. The fundamental mechanism is an iterative procedure that searches for the most appropriate placement solution (or solution hereafter in the text) based on the particle swarm optimization (PSO) strategy. As a preliminary step, we first calculate the average overflow probability for placing each VM on available PMs, and sort the VMs accordingly. We then select the most appropriate PM for each sorted VM, providing the metaheuristic with a good initial solution. During each iteration, we apply the particle updating rule to update the velocity and position of each particle in the swarm. To cope with the placement problem, we design a mapping operator to convert a particle in the continuous particle space

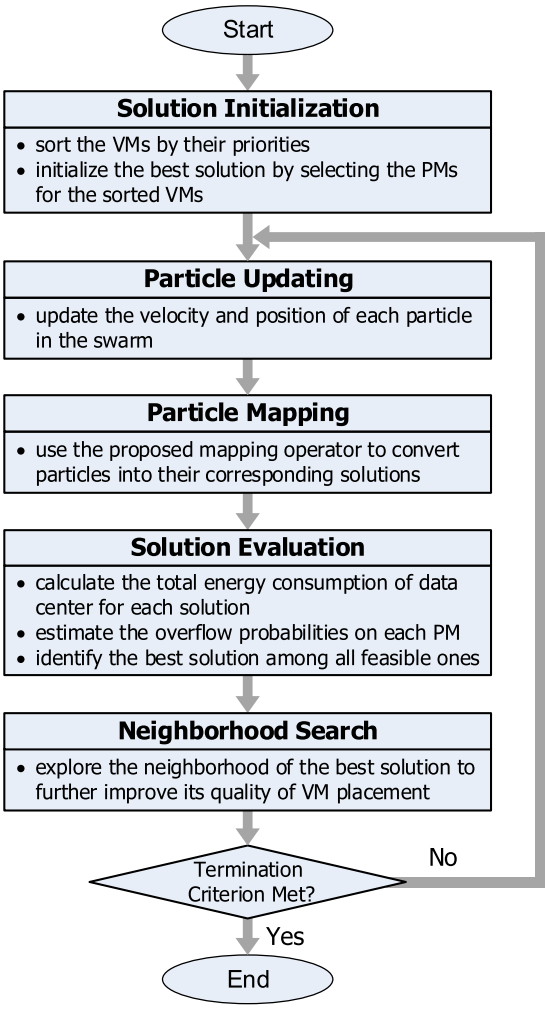


FIGURE 1. The flowchart of the stochastic VM placement algorithm.

into a candidate solution in the discrete solution space. For each converted solution, we use the estimation approach to evaluate its feasibility by predicting the corresponding CPU and memory overflow probabilities. Then, we identify the best solution with lowest energy cost among all feasible solutions. Furthermore, we employ a neighborhood search strategy to explore the neighborhood of the best solution to further improve its quality of VM placement. This procedure is iterated until the termination criterion is satisfied.

A. SOLUTION INITIALIZATION

A good initial solution is beneficial for enhancing the quality of the final solution produced by the proposed metaheuristic algorithm. For this reason, we develop a solution initialization method consisting of a VM prioritization step and a PM selection step. We first prioritize all VMs by their placement priorities that are calculated based on the average overflow probability on PMs. To be specific, the priority of VM i is determined as

$$\mathcal{P}_i = \frac{1}{k}\left(\sum_{j=1}^{k} \tilde{OF}_c^j\right) + \frac{1}{k}\left(\sum_{j=1}^{k} \tilde{OF}_m^j\right), \tag{20}$$

Algorithm 2 Solution Initialization

Input: uncertain CPU and memory requirements of each VM on each PM c_{ij} and m_{ij};
Output: an initial solution X_{init};
set $X_{\text{init}} \leftarrow [0, 0, \cdots, 0]^T$;
calculate the priority value for each VM to be placed according to Eq. (20);
sort the VMs in a descending order of priority values, and uses a list L to store the sorted VMs;
while *L is not empty* **do**
  select the first VM in list L;
  for $j = 1$ *to* k **do**
    check the available resource on PM j;
    if *PM j's resource cannot hold this VM* **then**
      continue;
    end
    calculate the energy overhead if the VM is placed on PM j;
  end
  assign the VM to the PM with lowest energy overhead and remove it from L;
  update the corresponding position in X_{init};
end
return X_{init};

where $\tilde{OF}_c^j$ and $\tilde{OF}_m^j$ denote the CPU and memory overflow probabilities by placing VM i on PM j, which can be calculated by Eqs. (17) and (19), respectively. Note that placing a specific VM on the designated PM would affect the PM's total resource utilization, and in turn the resource overflow probability on that PM. This VM priority in fact defines the extent that a VM may exceed the resource limit if it is placed. A higher VM priority indicates that the placement of this VM is more likely to cause resource overflow. Therefore, this VM has a higher priority when determining the uncertainty-aware placement solution.

In order to generate an initial solution to the stochastic placement problem, we sort all VMs in a descending order of the VM priority value defined in Eq. (20), and select the most appropriate PM for each VM. The algorithmic flow of solution initialization is summarized in Algorithm 2. We first set the initial solution as a zero vector. We use a list L to store the sorted VMs. For each VM in the list, we calculate the energy overhead by placing it on each PM, and select the PM with lowest energy overhead to hold this VM. The PMs that have insufficient resource to hold the VM will not be considered. This procedure is repeated until all VMs in the list have been placed. In this manner, a valid solution is obtained and will be used as the initial solution for the proposed metaheuristic.

B. PARTICLE MAPPING

According to the standard PSO framework [33], we assume a swarm consisting of N particles, and each particle is

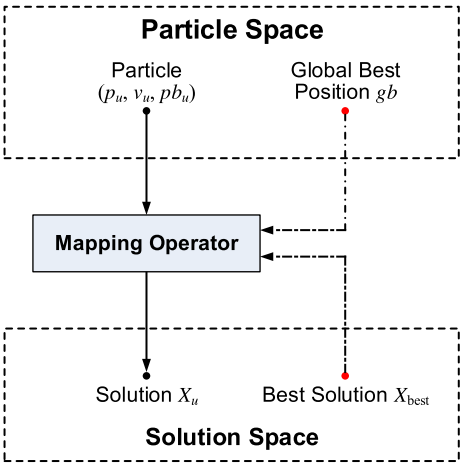


FIGURE 2. The mapping operator to convert a particle into a valid solution.

characterized by its position p_u and velocity v_u. In addition, pb_u is use to record the personal best position of each particle, and gb tracks the global best position among all particles. An important issue in the proposed algorithm is that, the particles are defined in a continuous space, whereas the solution space for this VM placement problem is discrete. To address this issue, we establish a mapping operator to convert a particle in the continuous particle space into a solution in the discrete solution space. Extended from the solution generation procedure in [34], the mapping operator generate offspring solution by combing the currently best solution and global best position gb in the swarm.

For a specific particle with position p_u, we use X_u to denote the placement solution that this particle can be mapped onto. We use X_{best} to represent the currently best solution explored during the search procedure. At the beginning of the metaheuristic algorithm, X_{best} is set as the initial solution obtained in Section 2. As illustrated in Figure 2, the mapping operator relies on both X_{best} in solution space and gb in particle space to implement the conversion from a particle to a solution. To start with, we initialize X_u as an empty set. The mapping operator works as follows to create a valid solution. Each particle is associated with a probability value $\mu \in [0, 1]$, and is compared with a given probability value ν. We then append an available PM to X_u in a sequential order according to the comparison between μ and ν. For each VM to be placed, if $\mu \leq \nu$ holds, we select the first available PM in X_{best} and append it to X_u. In the opposite situation that $\mu > \nu$, the PM to be appended to X_u will be determined by a probabilistic model that is dependent upon both X_{best} and gb. More specifically, we select the first available PMs in X_{best}, and use β to denote the number of remaining VMs in X_{best}. The g-th remaining VM ($g \leq \min\{\alpha, \beta\}$) will be appended to X_u with a probability P_{ug}, which is given by

$$P_{ug} = \frac{p_{ug} + gb_{ug}}{\sum_{l=1}^{\min\{\alpha,\beta\}} (p_{ul} + gb_{ul})}. \tag{21}$$

Algorithm 3 Particle Mapping

Input: i) a particle with position p_u;
ii) the global best solution gb;
iii) the currently best solution X_{best};
iv) a given probability ν;
Output: a placement solution X_u;

```
initialize X_u as an empty set;
if X_best does not exist then
    call Algorithm 2 to obtain X_init;
    set X_best ← X_init;
end
for i = 1 to n do
    generate a probability value μ at random;
    if μ ≤ ν then
        append the first PM in X_best to X_u;
    else
        select a PM according to the probability value in
        Eq. (21), and append this PM to X_u;
    end
end
return X_u;
```

After all VMs have been placed, a valid solution can be obtained. Algorithm 3 summarizes the algorithmic details of the mapping procedure.

C. SOLUTION EVALUATION

Given a candidate solution obtained by using the mapping operator in Algorithm 3, we need to evaluate the quality of this solution, including the optimization objective (i.e., the total energy of the data center) as well as the solution's feasibility (i.e., the resource overflow probabilities). As mentioned previously, the total energy consumption for a given placement solution can be predicted by summing over each PM's energy cost, which is further dependent on the CPU utilization of each VM this PM hosts. We assume that the CPU resources demanded by VMs are subject to normal distributions when placing VMs onto PMs. Under the uncertainty of resource requirements, we use the mean value of each VM's CPU requirement to calculate the energy costs of all PMs, as summarized in Algorithm 4. It is worth mentioning that, it is required to verify the feasibility of each candidate solution by evaluating the overflow probabilities. If the overflow probabilities for any solution exceed the threshold values, this solution is identified as an infeasible one and the corresponding total energy is set to a sufficiently large value.

D. NEIGHBORHOOD SEARCH

As is well known, general metaheuristic algorithms, including PSO algorithm, suffer from the limitation of easily being trapped into the first local optimum [35], [36]. To address this limitation, we further incorporate a neighborhood search strategy to improve the effectiveness of the

Algorithm 4 Solution Evaluation

Input: a placement solution X
Output: the total energy cost E_{Total}

determine A_{ij} values according to X;
determine B_j values according to A_{ij} values;
set $E_{\text{Total}} \leftarrow 0$;
for $j = 1$ *to* k **do**
  set $E_{\text{PM}} \leftarrow 0$;
  for $i = 1$ *to* n **do**
    $E_{\text{PM}} = E_{\text{PM}} + A_{ij} \cdot \lambda \cdot c_{ij}$;
  end
  if B_j *is true* **then**
    $E_{\text{PM}} = E_{\text{PM}} + E_{\text{stat},j}$;
  end
  $E_{\text{Total}} = E_{\text{Total}} + E_{\text{PM}}^{j}$;
  if $OF_c^{(j)} > \alpha_{\max}$ ***or*** $OF_m^{(j)} > \beta_{\max}$ **then**
  `/* overflow probability exceeds the threshold value */`
    $E_{\text{Total}} = \infty$; `/* a sufficiently large value */`
  end
end
return E_{Total};

Algorithm 5 Neighborhood Search

Input: the currently best solution X_{best};
Output: the best solution in the neighborhood $X_{\text{best}}^{\mathcal{N}}$;

set $X_{\text{best}}^{\mathcal{N}} \leftarrow X_{\text{best}}$;
for $i = 1$ *to* n **do**
  for $l = i + 1$ *to* n **do**
    swap x_i^* with x_l^* to form a new solution X'_{best};
    call Algorithm 1 to evaluate CPU and memory overflow probabilities for X'_{best};
    call Algorithm 4 to evaluate $E_{\text{Total}}(X'_{\text{best}})$;
    if *overflow probability exceeds the threshold value* **then**
      continue;
    end
    if $E_{Total}(X_{best})' < E_{Total}(X_{best}^{\mathcal{N}})$ **then**
      set $X_{\text{best}}^{\mathcal{N}} \leftarrow X'_{\text{best}}$;
    end
  end
end
update the currently best solution $X_{\text{best}} \leftarrow X_{\text{best}}^{\mathcal{N}}$;
return $X_{\text{best}}^{\mathcal{N}}$;

proposed metaheuristic by exploring the neighborhood of the best solution during each iteration.

As summarized in Algorithm 5, the first step of the neighborhood search strategy is to construct a set of neighborhood structures for each solution, in which we can search for better solutions. By evaluating all solutions that are converted from the particles, we can identify the currently best solution $X_{\text{best}} = [x_1^*, x_2^*, \cdots, x_n^*]$. We use a swap operation on any two elements in X_{best} to construct a neighborhood structure for X_{best}. In other words, swapping x_p^* with x_q^* denotes that the PM holding VM p and the PM holding VM q are interchanged. This swap operation results in a new placement solutions. For a given X_{best}, the total number of possible swaps is $n(n-1)/2$. Thus, we can determine a set of neighborhood structures $\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_r$, where $r = n(n-1)/2$. By evaluating the solutions in the neighborhood, it is possible to explore better solutions and help the proposed metaheuristic escape the local optimum. For each neighborhood solution, we first examine its feasibility by calling Algorithm 1 to predict the overflow probabilities. If the overflow probability exceeds the threshold value on any PM, this neighborhood solution is infeasible and will be neglected. We identify the best neighborhood solution among all feasible ones, and use it the replace the currently best solution X_{best}.

E. STOCHASTIC PLACEMENT ALGORITHM

This section details the proposed metaheuristic algorithm that seeks for the optimal solution to the stochastic VM placement problem. The details about this procedure is summarized in Algorithm 6. The metaheuristic starts with generating an initial solution, relying on which the swarm and particles can be initialized. Each particle in the swarm can be converted into a solution according to the mapping operator in Section V-B. We evaluate the feasibility as well as the quality of all converted solutions to determine the optimal one.

At the initialization step, we randomly initialize the values of particle position p_u and velocity v_u, and initialize each particle's pb_u value as its position p_u. The initial value of global best gb is determined by evaluating all particles in the swarm. By generating an initial solution and setting it as the currently best solution, we can use the mapping operator in Algorithms 3 to convert a particle into a valid solution, and can further evaluate the solution's energy efficiency as well as its feasibility. Among all feasible solutions satisfying the overflow probability constraints, the one with lowest energy cost is identified as gb of the swarm.

According to Algorithm 6, during each iteration, we first apply the following updating rule to produce a new set of particles [33]. Each particle's velocity v_u is updated by

$$v_u \leftarrow w \cdot v_u + c_1 \cdot r_1(p_u - pb_u) + c_2 \cdot r_2(p_u - gb). \quad (22)$$

The parameters r_1, r_2 and w are the structural parameters in PSO scheme (refer to [33] for details). The particle position is then updated by

$$p_u \leftarrow p_u + v_u. \quad (23)$$

Each updated particle is converted into a solution X'_u for evaluation. We perform Algorithm 4 to evaluate the total energy that solution X'_u yields. If X'_φ achieves better energy efficiency and satisfies the overflow probability constraints, we use X'_u to replace X_{best}, i.e., the best solution explored in

Algorithm 6 Stochastic VM Placement

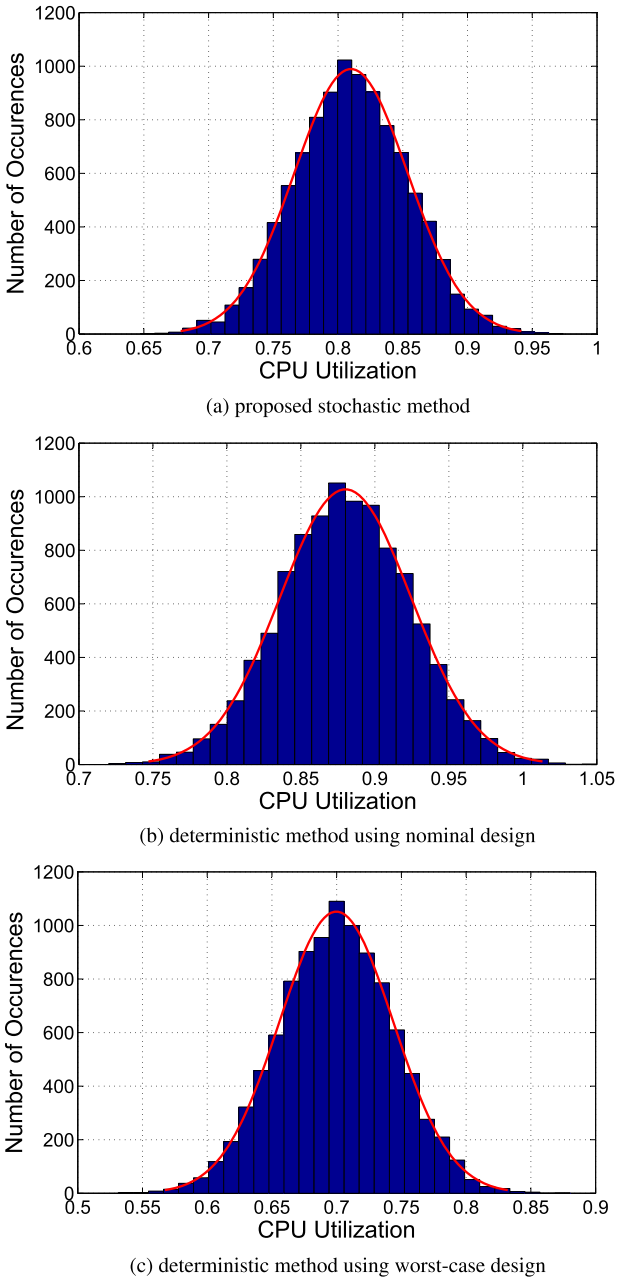
Input: i) uncertain CPU and memory requirements of each VM on each PM c_{ij} and m_{ij};
ii) user-specified threshold values $\alpha_{\max}$ and $\beta_{\max}$;
Output: the optimal solution X_{best} with lowest energy consumption

call Algorithm 2 to obtain X_{init} and set $X_{\text{best}} \leftarrow X_{\text{init}}$;
for $u = 1$ *to* P **do**
  initialize p_u and v_u randomly;
  set $pb_u \leftarrow p_u$;
  call Algorithm 3 to obtain a solution X_u;
  call Algorithm 4 to evaluate $E_{\text{Total}}(X_u)$;
end
determine gb among all particles;
while *termination criterion is not met* **do**
  use particle updating rule to update all particles;
  for $u = 1$ *to* P **do**
    call Algorithm 3 to obtain a solution X'_u;
    call Algorithm 1 to evaluate CPU and memory overflow probabilities OF_c and OF_m;
    call Algorithm 4 to evaluate $E_{\text{Total}}(X'_u)$;
    if $E_{Total}(X'_u) < E_{Total}(X_{best})$ **then**
      if $\forall OF_c^{(j)} \leq \alpha_{\max}$ ***and*** $\forall OF_m^{(j)} \leq \beta_{\max}$ **then**
        set $X_{\text{best}} \leftarrow X'_u$; /* update X_{best} if X'_u is feasible */
      end
    end
    update p_b for the particle;
  end
  update gb for the swarm;
  call Algorithm 5 to replace X_{best} by the best solution in its neighborhood;
end
return X_{best};

previous iteration. When all particles have been evaluated, we employ the neighborhood search strategy in Algorithm 5 to explore the neighborhood of X_{best} to further improve its quality. If a better solution is identified in the neighborhood of X_{best}, we replace X_{best} by this newly identified solution. At the end of each iteration, the personal best position for each particle and global best position for the swarm need to be updated according to Eqs. (22) and (23). When this iterative procedure satisfies the termination criterion, X_{best} is returned as the final solution to the stochastic VM placement problem.

VI. EVALUATION RESULTS

We use a set of heterogeneous server clusters to examine the effectiveness and energy efficiency of the proposed stochastic VM placement algorithm. We assume that the number of available PMs is at most 30 in the data center, and the number of VMs to be placed range from 100 to 300. The maximum amount of resource provided by the PM is set to 90%. The CPU and memory resources demanded by each



(a) proposed stochastic method

(b) deterministic method using nominal design

(c) deterministic method using worst-case design

FIGURE 3. Distributions of CPU utilization with a $\alpha_{\max}$ value of 0.02 by different VM placement methods.

VM are assumed to follow normal distributions. Considering the uncertainty of resource requirements, the threshold value for resource overflow probability is selected at different values for performance evaluation, ranging from 0.02 to 0.10.

A. EVALUATION OF SOLUTION FEASIBILITY

We first verify that the proposed stochastic placement approach can produce feasible solutions satisfying the probabilistic constraints on overflow probabilities for all PMs. For comparison purposes, we also consider the following two scenarios using deterministic VM placement in the presence of resource requirement variations.

TABLE 1. Overflow probabilities for different numbers of VMs and PMs.

Number of VMs	Number of PMs	Deterministic Method	Stochastic Method
100	10	32.67%	1.96%
	15	35.11%	1.97%
150	10	35.07%	1.95%
	15	33.66%	1.95%
200	15	34.60%	1.99%
	20	32.63%	1.97%
250	20	33.47%	1.99%
	25	34.95%	1.95%
300	25	34.58%	1.96%
	30	35.08%	1.96%

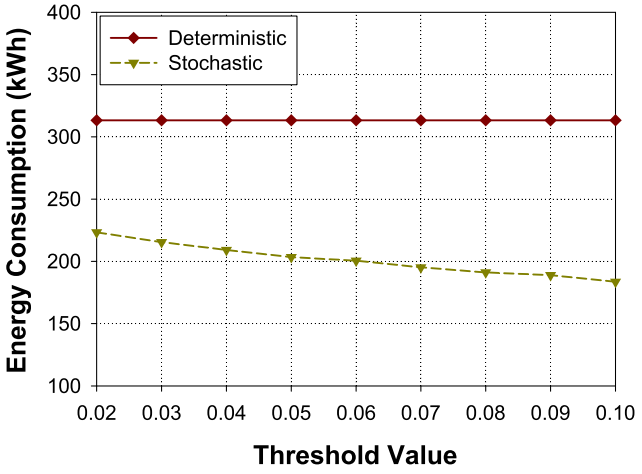


FIGURE 4. Energy consumptions for different threshold values.

- Nominal design: the mean values of uncertain CPU and memory requirements $\tilde{c}_{ij}$ and $\tilde{m}_{ij}$ are used in the VM placement framework to generate an uncertainty-unaware placement solution.
- Worst-case design: uncertain CPU and memory requirements are truncated at their 3σ values, and the maximum values are used in the VM placement framework to generate an uncertainty-unaware placement solution.

We first consider a configuration of 100 VMs and 10 PMs in the placement problem. The overflow probability threshold is set as 0.02, indicating a maximum overflow probability of 2% for each PM. We randomly generate 10,000 samples according to the solution obtained by the stochastic method. Fig. 3(a) shows the distribution of CPU utilization evaluated at all samples. We can observe that, the vast majority of the fluctuation values are below the overflow probability threshold value. More precisely, only 197 out 10,000 samples produce CPU utilization values that exceed 90%, indicating an overflow probability of 1.97%. The results by deterministic VM placement methods using nominal design and worse-case design are also presented in Figs. 3(b) and 3(c). Clearly, without consideration of resource requirement variations, the deterministic method using nominal design produces an overly optimistic solution that leads to a significant overflow probability of 32.67%. The deterministic method using worst-case design, on the other hand, leads to a conservative solution. Although this solution does not incur any overflow, a considerable energy overhead would be incurred due to the wastage of hardware resources, as will be justified in subsequent experiments.

We further verify the feasibility of placement solution for different configurations. Table 1 provides the overflow probabilities achieved by using deterministic and stochastic methods. We use a fixed threshold value of 2% for resource overflow probability. For various numbers of VMs and PMs, the deterministic method using nominal design always results in significant overflow probabilities, since it neglects the uncertainty of resource requirements. The maximum overflow probability reaches 35.11%. As mentioned previously, the conservative solutions by the deterministic method using worst-case design cause no overflows, and therefore the overflow probabilities are not provided in this table.

B. EVALUATION OF ENERGY EFFICIENCY

Having justified the feasibility of placement solution, we now verify the energy efficiency of the proposed stochastic method. Since the deterministic method using nominal design tends to produce infeasible solutions with high overflow probabilities, we only compare the energy efficiency between the deterministic method using worst-case design and the stochastic method. The comparison results in Table 2 show that, for different numbers of VMs and PMs, a reduction of power consumption can be achieved by the proposed stochastic method. Specifically, for the configuration of 200 VMs and 15 PMs, the power consumption by the deterministic method is 353.48 kWh, while the power consumption by the stochastic method has been reduced to 248.44 kWh, indicating a maximum energy reduction of 29.72%. The average energy reduction for all configurations is 26.74%. This improvement of energy efficiency is due to the fact that the conservative solutions generated by the deterministic method using worst-case design would cause extra energy overhead for turning on more PMs.

We further evaluate the energy efficiency of the stochastic method by selecting different threshold values for overflow probabilities. Figure 4 presents the energy consumption by the stochastic method and deterministic method, respectively. The number of VMs is fixed at 200, and the number of PMs is fixed at 20. For different threshold values, the stochastic method is capable of producing feasible placement solutions that satisfy the overflow probability constraint under the impact of uncertain resource requirements. More importantly, as the overflow probability threshold value increases, the improvement of energy efficiency achieved by the stochastic method becomes more significant.

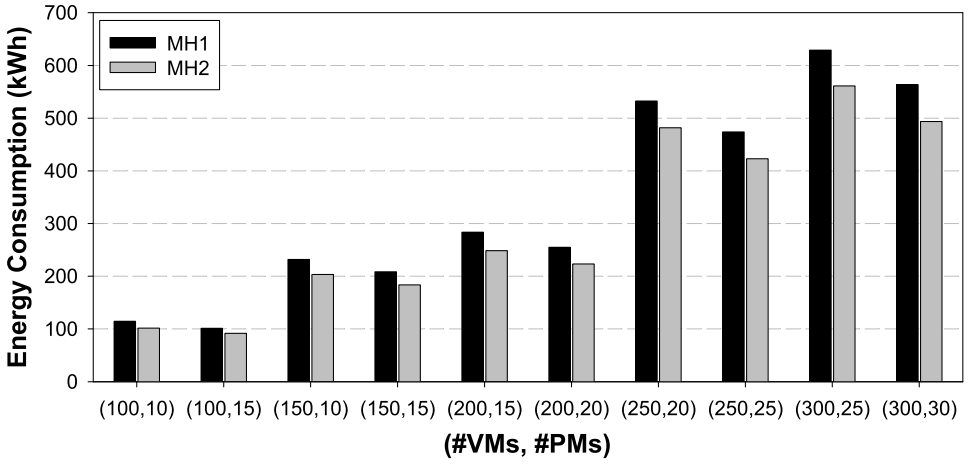


FIGURE 5. Energy consumptions by MH1 and MH2 for different configurations.

TABLE 2. Energy consumptions (in kWh) for different numbers of VMs and PMs.

Number of VMs	Number of PMs	Deterministic Method	Stochastic Method	Energy Reduction
100	10	139.10	101.70	26.89%
	15	124.70	91.75	26.42%
150	10	278.71	203.42	27.01%
	15	249.14	183.64	26.29%
200	15	353.48	248.44	29.72%
	20	313.30	223.28	28.73%
250	20	651.03	481.83	25.99%
	25	585.28	423.20	27.69%
300	25	753.36	561.11	25.52%
	30	663.48	493.63	25.60%

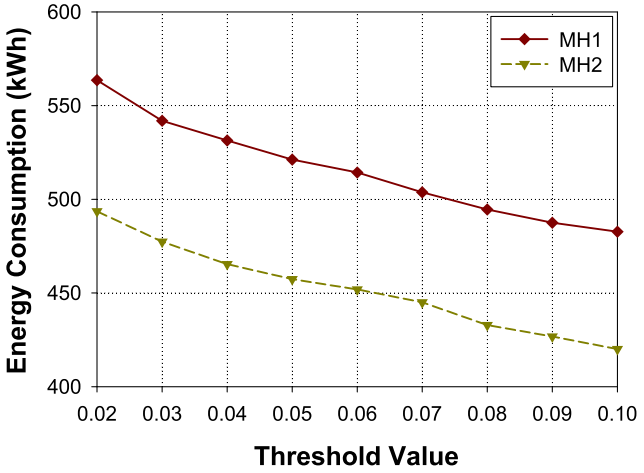


FIGURE 6. Energy consumptions by MH1 and MH2 for different threshold values.

When the threshold value reaches 10%, an energy reduction of 41.41% can be observed. The results indicate that, by relaxing the probabilistic constraint on overflow probability, the stochastic method can explore more energy-efficient placement solutions.

C. EVALUATION OF KEY STRATEGIES

In this work, we incorporate a solution initialization procedure and a neighborhood search strategy into the proposed metaheuristic. We now investigate the impacts of these key strategies upon the placement results. We use MH1 and MH2 to denote the metaheuristics with and without the two strategies, respectively. Note that MH2 is in fact the metaheuristic algorithm in the preliminary version of this work [16].

We first compare the energy consumptions for the placement solutions obtained by using MH1 and MH2. To fully justify the performance of metaheuristics, we repeat each set of experiments 10 times to evaluate the energy consumption corresponding to the placement solution, and use the average values for comparison. The results in Figure 5 shows that, MH2 outperforms MH1 in terms of reduced energy consumption for various configurations with different numbers of VMs and PMs. The energy reductions ranges from 9.36% to 12.46%. This observation is due to the fact that, the solution initialization and neighborhood search strategies in MH2 enhances the searching ability of the metaheuristic, and in turn the energy efficiency of the proposed metaheuristic.

The same conclusion can be drawn by changing the threshold values for overflow probabilities. Compared with MH1, the placement solutions produced by MH2 reduce the energy consumption to different extents. When the threshold value is set as 10%, the energy reduction reaches 12.98%. According to the analysis in Section V, the solution initialization procedure in MH2 in fact provides the metaheuristic algorithm with a good initial solution, and the neighborhood search strategy can identify better solutions in the neighborhood of the best solution obtained during each iteration.

VII. CONCLUSION

This paper proposes a stochastic VM placement algorithm for energy-efficient data centers under resource requirement variations. The uncertainty of resource requirements demanded by the VMs are modeled as random variables, and the uncertainty-aware VM placement problem is formulated as a stochastic optimization problem that imposes a probabilistic constraint on resource overflow probability. A metaheuristic algorithm to determine the most appropriate solution that minimizes the total energy consumption under the probabilistic resource constraint. A solution initialization procedure and a neighborhood search strategy are incorporated to further improve the quality of placement solution. Experimental results have justified the feasibility and energy efficiency of the placement solution obtained by the stochastic algorithm.

ACKNOWLEDGMENT

This article was presented in part at the Proceedings of International Conference on Cyber, Physical and Social Computing (CPSCOM), Atlanta, GA, USA, July 14–16, 2019.

REFERENCES

[1] I. Pietri and R. Sakellariou, "Mapping virtual machines onto physical machines in cloud computing: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, p. 49, 2016.

[2] Y. Zhang and J. Sun, "Novel efficient particle swarm optimization algorithms for solving QoS-demanded bag-of-tasks scheduling problems with profit maximization on hybrid clouds," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 21, p. e4249, 2017.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[4] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu, and R. Buyya, "Energy-aware virtual machine allocation for cloud with resource reservation," *J. Syst. Softw.*, vol. 147, pp. 147–161, Jan. 2019.

[5] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, May 2016.

[6] Z. Á. Mann, "Multicore-aware virtual machine placement in cloud data centers," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3357–3369, Nov. 2016.

[7] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. San Rafael, CA, USA: Morgan & Claypool, 2018.

[8] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, Jun. 2007, pp. 13–23.

[9] Z. Tang, Y. Mo, K. Li, and K. Li, "Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment," *J. Supercomput.*, vol. 70, no. 3, pp. 1279–1296, 2014.

[10] T. Duong-Ba, T. Nguyen, B. Bose, and T. Tran, "Joint virtual machine placement and migration scheme for datacenters," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 2320–2325.

[11] T. Yang, Y. C. Lee, and A. Y. Zomaya, "Energy-efficient data center networks planning with virtual machine placement and traffic configuration," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2015, pp. 284–291.

[12] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, "Statistical profiling-based techniques for effective power provisioning in data centers," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, Apr. 2009, pp. 317–330.

[13] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proc. 7th Int. Conf. Autonomic Comput.*, Jun. 2010, pp. 11–20.

[14] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," in *Proc. 2nd ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst.*, Mar. 2007, pp. 289–302.

[15] B. B. Nandi, A. Banerjee, S. C. Ghosh, and N. Banerjee, "Stochastic VM multiplexing for datacenter consolidation," in *Proc. IEEE 9th Int. Conf. Services Comput.*, Jun. 2012, pp. 114–121.

[16] S. Yan, Y. Zhang, S. Tao, X. Li, and J. Sun, "A stochastic virtual machine placement algorithm for energy-efficient cyber-physical cloud systems," in *Proc. IEEE Cyber, Phys. Social Comput.*, Jul. 2019, pp. 587–594.

[17] C. Isci, J. E. Hanson, I. Whalley, M. Steinder, and J. O. Kephart, "Runtime demand estimation for effective dynamic resource management," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2010, pp. 381–388.

[18] M. Sindelar, R. K. Sitaraman, and P. Shenoy, "Sharing-aware algorithms for virtual machine colocation," in *Proc. 23rd Annu. ACM Symp. Parallelism Algorithms Archit.*, Jun. 2011, pp. 367–378.

[19] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," *Future Gener. Comput. Syst.*, vol. 93, pp. 278–289, Apr. 2019.

[20] Y. Chen, X. Chen, W. Liu, Y. Zhou, A. Y. Zomaya, R. Ranjan, and S. Hu, "Stochastic scheduling for variation-aware virtual machine placement in a cloud computing CPS," *Future Gener. Comput. Syst.*, to be published, doi: 10.1016/j.future.2017.09.024.

[21] G. Wu, M. Tang, Y.-C. Tian, and W. Li, "Energy-efficient virtual machine placement in data centers by genetic algorithm," in *Proc. Int. Conf. Neural Inf. Process.*, 2012, pp. 315–323.

[22] Z. Yang, L. Liu, S. Das, R. Ramesh, A. Y. Du, and C. Qiao, "Availability-aware energy-efficient virtual machine placement," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 5853–5858.

[23] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware VM placement for cloud systems," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2012, pp. 498–506.

[24] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. Int. Conf. Inf. Commun.*, Apr. 2011, pp. 71–75.

[25] X. Dai, J. M. Wang, and B. Bensaou, "Energy-efficient virtual machine placement in data centers with heterogeneous requirements," in *Proc. IEEE 3rd Int. Conf. Cloud Netw.*, Oct. 2014, pp. 161–166.

[26] R. Kanagavelu, B.-S. Lee, N. T. D. Le, L. N. Mingjie, and K. M. M. Aung, "Virtual machine placement with two-path traffic routing for reduced congestion in data center networks," *Comput. Commun.*, vol. 53, pp. 1–12, Nov. 2014.

[27] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proc. Int. Conf. Parallel Distrib. Syst.*, Dec. 2013, pp. 102–109.

[28] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput.*, Dec. 2010, pp. 179–188.

[29] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, 2013.

[30] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Dec. 2009, pp. 103–110.

[31] R. Saini, "A multi-objective ant colony system algorithm for virtual machine placement," *Int. J. Eng. Res. Appl.*, vol. 7, no. 1, pp. 95–97, 2017.

[32] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York City, NY, USA: McGraw-Hill, 2012.

[33] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov. 1995, pp. 1942–1948.

[34] C. Rajendran and H. Ziegler, "Two ant-colony algorithms for minimizing total flowtime in permutation flowshops," *Comput. Ind. Eng.*, vol. 48, no. 4, pp. 789–797, 2005.

[35] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[36] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, 2001.

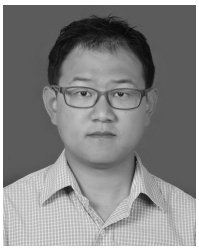
JUNLONG ZHOU (S'15–M'17) received the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, from 2014 to 2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include real-time embedded systems, cloud computing, and cyber physical systems. He has published 50 refereed articles in his research areas, most of which are published in premium conferences and journals, including the ACM/IEEE DATE, the IEEE TC, the IEEE TPDS, the IEEE TCAD, the IEEE TCAS, the IEEE TR, the IEEE TAES, and so on. He received the Reviewer Award from the Journal of Circuits, Systems, and Computers, in 2016. He has been an Associate Editor for the *Journal of Circuits, Systems, and Computers*, and serves as a Guest Editor for several special issues of the *ACM Transactions on Cyber-Physical Systems*, the *IET Cyber-Physical Systems: Theory & Applications*, and the *Journal of Systems Architecture* (Elsevier).

YI ZHANG received the B.S. and Ph.D. degrees from the School of Computer Science and Engineering, Southeast University, Nanjing, China, in 2005 and 2011, respectively. From July 2009 to December 2009, he did an internship at the IBM China Research Laboratory after he was awarded the IBM Ph.D. Fellowship. In 2011, he joined the Huawei Tech. Company, as a member of the Technical Research Staff. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include project scheduling, workflow optimization, resource management, and allocation in cloud computing and mobile computing.

LULU SUN received the B.S. degree in software engineering from Nanjing University of Science and Technology, Nanjing, China, in 2017. She is currently working toward the M.S. degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her research interests include cloud computing and task scheduling.

SISI ZHUANG is currently pursuing the B.S. degree in intelligent science and technology with the Nanjing University of Science and Technology, Nanjing, China. Her research interests include cloud computing and task scheduling.

CHENG TANG is currently pursuing the B.S. degree in network engineering with the Nanjing University of Science and Technology, Nanjing, China. His research interests include cloud computing and task scheduling.



JIN SUN (M'17) received the B.S. and M.S. degrees in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Arizona, in 2011. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include cloud and edge computing, stochastic modeling and analysis, and cyber-physical systems.