Applied Intelligence (2019) 49:220–232
https://doi.org/10.1007/s10489-018-1264-5

# Chemical reaction optimization for virtual machine placement in cloud computing

Zhiyong Li[1,2] · Yang Li[1,2] · Tingkun Yuan[1,2] · Shaomiao Chen[1,2] · Shilong Jiang[3]

Published online: 24 August 2018

**Abstract**
With the development of virtualization technologies, cloud data centers are faced with more and more virtual machines (VMs) requests. How to realize an efficient virtual machine placement (VMP) becomes a hot research topic. The optimal resource consumption and the utilization rate of a physical machine in the whole cloud data center can be realized by optimizing the process from the virtual machine to the physical machine. VMP is a combinatorial optimization problem which was demonstrated to be NP-hard. In this paper, a new formulation of VMP problem is presented by taking into consideration the optimization of the two following objectives: (i) minimize the energy consumption, and (ii) maximize the resource utilization. In order to achieve these targets, we propose two algorithms based on chemical reaction optimization (CRO) algorithm, namely CVP and CVV, with two types of solution representation. The proposed algorithms are compared with other optimal placement strategies, namely Cuckoo Search Optimization (CSO), Reordered Grouping Genetic Algorithm (RGGA), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD). Experimental results show that the proposed CVP and CVV give better performance comparing with the other compared algorithms in terms of resource consumption and resource utilization. In term of scalability, the proposed CVV algorithm benefits from the high computational speed and performs well when there are a large number of virtual machine scheduling requests in the cloud data center.

**Keywords** Cloud computing · Virtual machine placement · Energy consumption · Resource utilization · Chemical reaction optimization

## 1 Introduction

As a new computing model [1, 2], cloud computing has attracted wide attention from the academia and industry communities. Cloud computing can provide the computing resources in an on-demand manner [3, 4] and similar to public utility services, it can be bought via a pay-as-you-go model [5]. There are three major types of services facilitated by cloud, namely infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [6]. In the IaaS layer, the services provided by the cloud data centers to consumers aim to the utilization of all facilities, including CPU, RAM, network and the other basic computing resources. The proposed work mainly focuses on how to efficiently implement the scheduling algorithm of placing virtual machines on physical machines in the cloud data centers so as to minimize resource consumption and maximize resource utilization [32]. This kind of problem is called the virtual machine optimal placement problem. In fact, the virtual machine placement (VMP) problem can be modeled as a Multi-Packing Problem (MCBPP), which is one of the most popular NP-Hard combinatorial optimization problems. Besides, there are different sized items to be packed into a box of a given capacity. the main objective is to minimize the number of the used boxes. In our case, the items to be packaged can be represented by the VMs and their sizes are represented by the resource utilization. Moreover, the bins are the PMs and their capabilities are the multi-dimensional resources such as CPU, RAM, etc [33].

✉ Zhiyong Li
zhiyong.li@hnu.edu.cn

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

[2] Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha, China

[3] PKU-HKUST Shenzhen-HongKong Institution, Shenzhen, China

An efficient placement work is strongly depends on the virtualization technology. The virtualization enables the sharing the of the computational resources by virtualizing the physical resources as multiple isolated execution environments [35, 39]. Virtualization provides a promising approach in which one or more machines on the hardware resources can be through the partial or complete machine to simulate the time-sharing. Hardware and software is divided into a multiple execution environment, each one can be used as a complete system [36]. This technique also supports on-demand or practical calculation model, a real-time resource supply. Moreover, in this model, the computing resources are only related to the need provided to the application rather than statically allocated based on peak workload demand. Through virtualization, cloud providers can ensure that service quality (QoS) is delivered to users and achieving higher resource utilization and efficient energy consumption [34].

Chemical reaction optimization (CRO) is a new meta-heuristic algorithm proposed in [18] and inspired from the behavior of the chemical reaction processes. Since its appearance, the CRO algorithm has been successfully applied to solve many kinds of problems. This includes the combinatorial optimization problems such as the channel assignment problem in wireless mesh networks, the 0-1 knapsack problem [19], task scheduling in grid computing [20], the flexible job-shop scheduling problems with maintenance activity constraints [22], the target recognition in aerial images [23]. As well as the global numerical optimization problems [21] and the standard continuous benchmark functions [24].

In this paper, two new CRO-based algorithms, namely CRO-VMP-Permutation (CVP) and CRO-VMP-Vector (CVV), are developed for solving the VMP problem with two different representations. The proposed algorithms aim to achieve both objectives: higher resource utilization as well as the optimal energy consumption [42]. Besides, a new scheduling strategy is implemented to achieve the afore-mentioned goals. Moreover, the implemented scheduling strategy is as scalable as possible which allows the algorithms to efficiently work in case of the large-scale cloud computing platforms.

### 1.1 Paper contribution

The main contributions of this work can be summarized as follows:

1. Based on the CRO framework, two different CRO based algorithms, CVP and CVV, are proposed to complete the task of scheduling the virtual machines to physical machines.
2. CVP and CVV are superior than FFD and BFD in terms of energy consumption and resource utilization. Moreover, Comparing with CSO and RGGA, CVP and CVV have a comprehensive advantage in terms of energy consumption in most cases. As well as, CVP and CVV have a certain advantage in resource utilization.
3. At the same time, CVV has strong scalability relative to CVP, this means that CVV can be applied to resource scheduling tasks for large-scale cloud computing platforms.

### 1.2 Paper structure

The rest of this paper is organized as follows. Section 2 presents a review of the VMP related works. In Section 3, the VMP problem's formulation is given. The framework of two proposed CRO based algorithms with the proposed representations are described in Section 4. The simulation results are discussed in Section 5. Finally, Section 6 concludes the paper and gives some perspectives for future works.

## 2 Related work

An efficient scheduling algorithm does not only ensure the QoS, but it also has a higher resource utilization and low energy consumption. With the rapid growth of cloud computing centers, physical machine resource consumption has increased to a high level [9] and inefficient resource utilization has resulted in resource waste of 11% and 50% [10]. Therefore, how to reduce the resource consumption of physical machines and how to improve the resource utilization of the physical machines has become a hot issue in the field of cloud computing where the main objective is to ensure the existing QoS. This involves the VMP problem which is a combinatorial optimization problem that aims to define an optimal strategy to place the virtual machines on the appropriate physical machines [11]. VMP problem is often formulated as a variant of the vector bin-packing problem which is known as an NP-hard optimization problem [12]. For the cost control of the cloud providers, reducing the energy consumption is considered first. Moreover, a good VMP solution could effectively improve the resources utilization, reduce the number of physical machines used and eventually reduce the energy consumption.

In past studies, several methods have been developed to solve VMP problem beginning from the exact algorithms to more intelligent meta-heuristic algorithms. For the exact algorithms, First fit decreasing (FFD) [29] method is a common method which has been applied to find a solution to the vector bin packing problem. Later, FFD has also been applied to solve the VMP problem. Best fit decreasing (BFD) [30] algorithm is a best known algorithm for online bin packing and it can be considered a good algorithm for VMP problem in cloud environments. The meta-heuristic algorithms [33, 37] are also used to host the VMs by optimizing the assignment of VMs [8, 13]. For instance, in [36], a Levy based whale optimization algorithm is proposed to solve the optimal utilization allocation problem. In [8], Sait et al. designed a multi-objective CSO algorithm to simultaneously optimize the power consumption and resource wastage of the datacenter. In the same context, Wilcox et al. [38] proposed the reordering packet genetic algorithm (RGGA) which achieved good results on solving the Multi-capacity Bin Packing Problem (MCBPP).

In contrast with the previous works, other works have been proposed which aim to maximize the resource utilization ratio of the physical servers through the VM consolidation [14]. In [31], Lopez-Pires et al. have proposed a Two-Phase optimization schemes for the VMP problem. Besides, the optimization goal is the higher resource utilization. In [15, 16], a load balancing on different physical servers is created in order to improve the overall system efficiency. Sait et al. [17], the authors have developed a goodness measure to evaluate the placement performance. In this paper, a new virtual machine scheduling algorithm is designed to minimize the total energy consumption and maximize the resource utilization. Moreover, in order to deal with the large-scale cluster, another performance metric is evaluated in order to test the scalability of the proposed algorithms in addition with the above two optimization objectives.

## 3 VMP problem

### 3.1 Power consumption modeling

As known, the energy consumed by a physical machine comes principally from the CPU, RAM, storage systems and so on. From recent studies, we summarize that the power consumption of physical machines can be accurately described by a linear relationship between the power consumption and the CPU utilization [25]. To save energy, unused physical machines are often shut down, so their power consumption in idle state is not part of the total energy consumed by the CPU. Hence, we define the power consumption as a function of the CPU utilization as follows:

$$p_j = \begin{cases} \left[ (p_j^{busy} - p_j^{idle}) \times R_j^{cpu} \right] + p_j^{idle} & if\ R_j^{cpu} > 0 \\ 0 & otherwise \end{cases} \tag{1}$$

where $p_j^{busy}$ and $p_j^{idle}$ are the average power values when the $pm_j$ is fully utilized and is idle respectively. $R_j^{cpu}$ is the CPU utilization of $pm_j$. Real data shows that $p_j^{idle}$ is around $0.7 \times p_j^{busy}$. Refer to the parameter setting of [8], we fix the values to of $p_j^{busy}$ and $p_j^{idle}$ to 215 and 162 Watt.

### 3.2 Problem formulation

VMP is the process of mapping a set of virtual machines to a set of physical machines. We assume that the set of virtual machines, that may be requested from one or more than one tenant, need to be placed on a set of physical machines [37, 42]. Let $n$ and $m$ denote the set size of $VMs$ and $PMs$ respectively. The resources required usually are the CPU, RAM, disk space, bandwidth and so on [31, 36]. Here we only consider the two resources CPU and RAM. We suppose that the CPU and RAM resource requests of $vm_i$ are $(vm_i^{cpu}, vm_i^{ram})$, $i = 1, 2, ...n$. The CPU and RAM resources offered by $pm_j$ are $(pm_j^{cpu}, pm_j^{ram})$, $j = 1, 2...m$. Two binary variables $map_{ij}$ indicates that $vm_i$ is assigned to $pm_j$ and the binary variable $y_j$ indicates whether $pm_j$ is used or not [17], where

$$y_j = \begin{cases} 1 & if\ pm_j\ in\ use \\ 0 & otherwise \end{cases}$$

In order to save the energy, physical machines that are idle will be turned to sleep mode or off [8]. Thus, their idle power is not considered as a part of the total energy consumption. Moreover, since the objective is to minimize the total energy consumption, the placement problem can be formulated as:

$$min.\ \ f(\mathbf{y}) = \sum_{j=1}^{m} y_j \times p_j \tag{2}$$

Subject to:

$$\sum_{i=1}^{n} (vm_i^{cpu} \times map_{ij}) < pm_j^{cpu} \tag{3}$$

$$\sum_{i=1}^{n} (vm_i^{ram} \times map_{ij}) < pm_j^{ram} \tag{4}$$

$$\sum_{j=1}^{m} map_{ij} = 1 \tag{5}$$

$$y_i, map_{ij} \in \{0,\ 1\} \qquad i = 1, 2...n \qquad j = 1, 2...m$$

where (3) and (4) ensure that the sum of resources requirement of VMs placed onto the corresponding physical machine should be less than the offered resources. Equation (5) guarantees that a virtual machine can only be placed onto a physical machine at the same time. Moreover, we not only deal with the power consumption issue but we take also into consideration the improvement of the resource utilization [42]. The $pm_j$ resource utilization is defined as $R_j = a \times R_j^{cpu} + b \times R_j^{ram}$. Hence, the average resource utilization of all used physical machines can be formulated as follows:

$$R_j^{cpu} = \frac{\sum_{i=1}^{n} (map_{ij} \times vm_i^{cpu})}{pm_j^{cpu}} \tag{6}$$

$$R_j^{ram} = \frac{\sum_{i=1}^{n} (map_{ij} \times vm_i^{ram})}{pm_j^{ram}} \tag{7}$$

$$\bar{R} = \frac{\sum_{j=1}^{m} (a \times R_j^{cpu} + b \times R_j^{ram})}{\sum_{j=1}^{m} y_j} \tag{8}$$

where $a$ and $b$ are the weight of the CPU and RAM respectively, $a + b = 1$ and $0 \leq a, b \leq 1$. Without loss of generality, we take $a = b = 0.5$.

For the convenience of the reader, the symbols defined above are summarized in Table 1.

## 4 Proposed meta-heuristic algorithms

### 4.1 CRO solves the VMP problem

In this paper, we designed the permutation-based and vector-based representations for the VMP problem. They are named as CVP and CVV respectively.

For the CRO-VMP-Permutation, one vector is usually used which is a permutation of 1, 2, ...$n$ and the value of the element is the virtual machine's ID and the position of the element is the virtual machine's order placed by First Fit (FF). FF means that the virtual machine is placed into the first physical machine that has enough resource. Suppose that we have ten virtual machines to be placed on six physical machines, a possible solution can be represented as $\boldsymbol{\omega} = [1, 10, 2, 6, 3, 8, 4, 7, 5, 9]$. This means that the first virtual machine is placed firstly, the tenth virtual machine is placed secondly, the second virtual machine is placed thirdly and so on. The left subgraph of Fig. 1 shows an example of the CVP placement policy. Besides, ten VMs are placed into six PMs with two dimensions, i.e., CPU and RAM. The parameter $\boldsymbol{y}$ in problem (2) is $\boldsymbol{y} = [1, 1, 1, 1, 1, 0]$.

Table 1 Main symbols

| Symbols | Description |
|---|---|
| $vm_i$ | a virtual machine in $VM$ |
| $pm_j$ | a physical machine in $VM$ |
| $vm_i^{cpu}$ | the CPU requirement of $vm_i$ |
| $vm_i^{ram}$ | the RAM requirement of $vm_i$ |
| $pm_j^{cpu}$ | the CPU capacity of $pm_j$ |
| $pm_j^{ram}$ | he RAM capacity of $pm_j$ |
| $R_j^{cpu}$ | the CPU utilization on $pm_j$ |
| $R_j^{ram}$ | the RAM utilization on $pm_j$ |
| $map_{ij}$ | a binary variable indicates virtual machine $vm_i$ assigned to physical machine $pm_j$ |
| $y_j$ | a binary variable indicates whether $pm_j$ is in use or not |

For the CRO-VMP-Vector, the number of the elements in the vector is equal to the number of the virtual machines to be placed. The position of the element in the vector denotes the virtual machine's ID, all the elements are integers in the range of $[1, m]$, where m is the total number of physical machines. The value of an element indicates which physical machine is chosen to place the corresponding virtual machine. Note that, these these values can be repetitive since a physical machine can contain more than one virtual machine.

For example, suppose ten virtual machines are placed on six physical machines. More precisely, the first, third and tenth virtual machine are placed on physical machine 1. The second, fourth and sixth virtual machine are placed on physical machine 2. The fifth virtual machine is placed on physical machine 3. Finally, the rest are placed on physical machine 4. Using the vector-based representation, the vector solution $\boldsymbol{\omega}$ can be represented by $\boldsymbol{\omega} = [1, 2, 1, 2, 3, 2, 4, 4, 4, 1]$. However, it is worth noticing that this technique may produce an infeasible solution which violates the constraint relations (4)–(6). Therefore, a repair operator is needed to transform the unfeasible solutions to feasible ones. The right subgraph of Fig. 1 shows an example of the CVV placement policy. Besides, ten VMs are placed on six PMs with two dimensions, i.e., CPU and RAM. The parameter $\boldsymbol{y}$ in problem (2) is $\boldsymbol{y} = [1, 1, 1, 1, 0, 0]$.

As mentioned above, CRO-VMP-Permutation-based and CRO-VMP-Vector-based representations are used for CVP and CVV algorithms respectively. In fact, the proposed algorithms not only differ on the representation of the solution, but also the generation of new solutions using the CRO operators are completely different. In the next two subsections, we will describe the different operators for the proposed algorithms CVP and CVV.

### 4.2 CVP

As indicated before, using the permutation-based technique, we usually need to use one vector in order to represent the solutions. The CVP flowchart is given in Fig. 2.

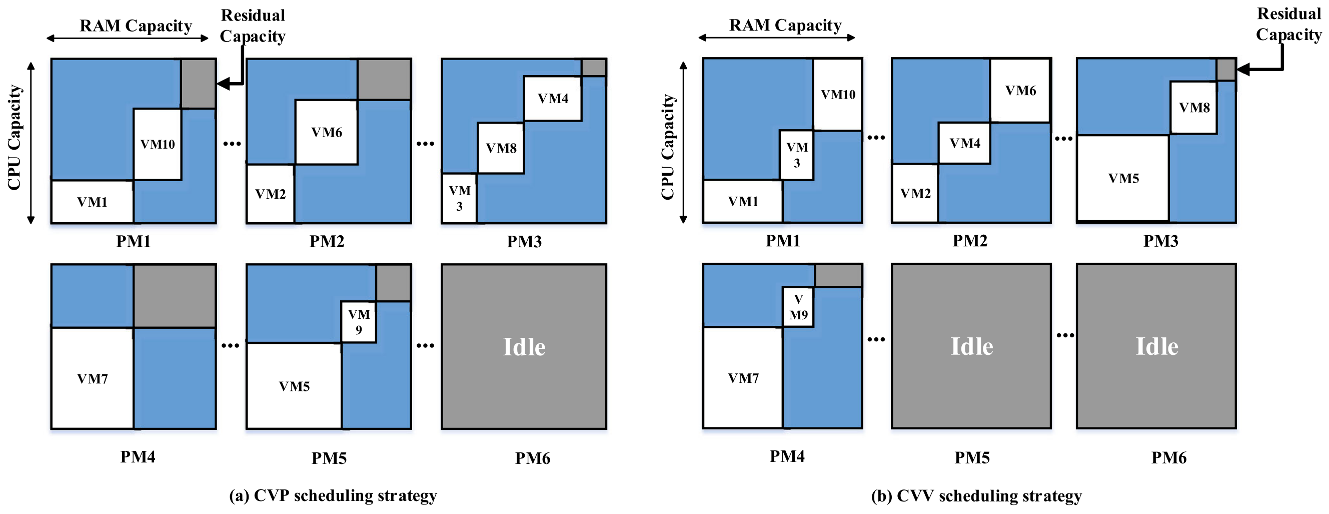
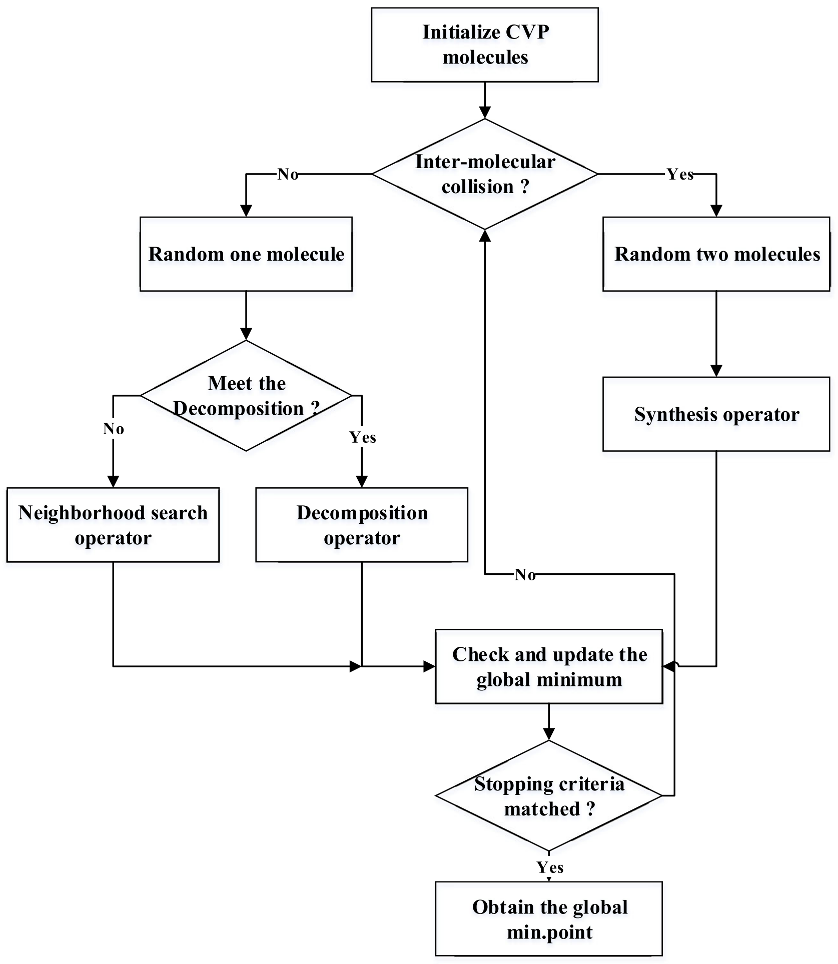


**Fig. 1** Two virtual machine placement strategies examples based on CRO

### 4.2.1 Neighborhood search operator

This operator is used for the on-wall ineffective collision and the inter-molecular ineffective collision to improve both the diversification and the intensification of the algorithm. It is used to look for new neighbor solutions located near from the current one. In our proposed neighborhood search operator for the CVP algorithm, after the selection of one

**Fig. 2** Flowchart of CVP


Initialize CVP molecules
Inter-molecular collision ?
No
Yes
Random one molecule
Random two molecules
Meet the Decomposition ?
No
Yes
Synthesis operator
Neighborhood search operator
Decomposition operator
No
Check and update the global minimum
Stopping criteria matched ?
Yes
Obtain the global min.point

solution, we select randomly two numbers and we exchange their corresponding positions. In the following example, the positions of 4 and 7 are exchanged.

$$[1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10] \rightarrow [1,\ 2,\ 3,\ 7,\ 5,\ 6,\ 4,\ 8,\ 9,\ 10]$$

### 4.2.2 Decomposition operator

Generally, the decomposition operator is used to improve the diversity on the algorithm. In our proposed CVP decomposition operator, we adopt the circular shift operator to produce two new solutions $\boldsymbol{\omega}'_1$ and $\boldsymbol{\omega}'_2$ from the current solution $\boldsymbol{\omega}$. Firstly, two not equal integers are generated in the range of $[-n, n]$ where n is the size of the solution. Negative integer means shift to the left and positive integer means shift to the right. In the following example, with the values 2 and 3 are generated, we will have:

before decomposition

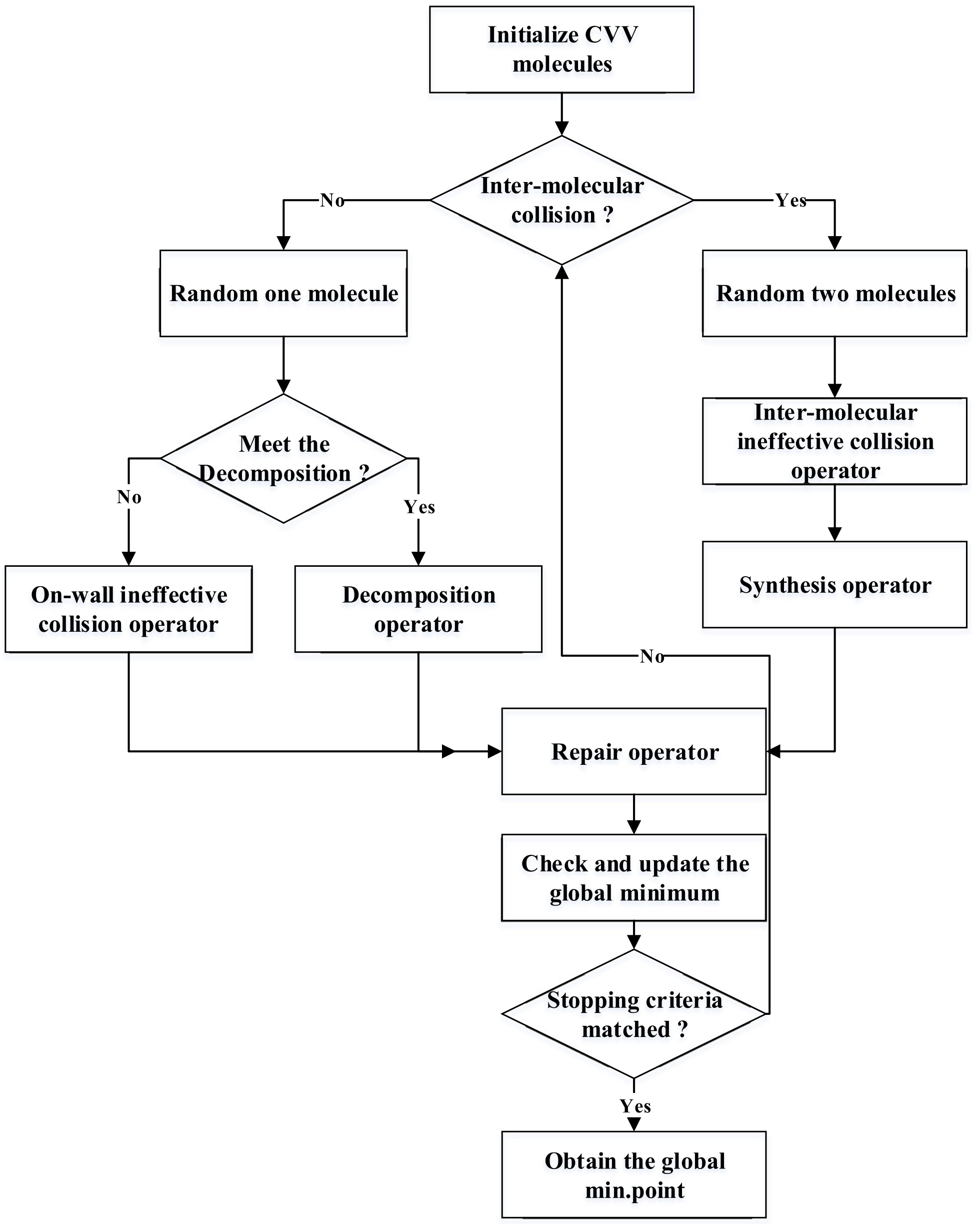
$$\boldsymbol{\omega} : [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10]$$

after decomposition

$$\boldsymbol{\omega}'_1 : [3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 1,\ 2]$$

$$\boldsymbol{\omega}'_2 : [8,\ 9,\ 10,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7]$$

**Fig. 3** Flowchart of CVV


Initialize CVV molecules
Inter-molecular collision ?
No
Yes
Random one molecule
Random two molecules
Meet the Decomposition ?
No
Yes
Inter-molecular ineffective collision operator
Synthesis operator
On-wall ineffective collision operator
Decomposition operator
No
Repair operator
Check and update the global minimum
Stopping criteria matched ?
Yes
Obtain the global min.point

### 4.2.3 Synthesis operator

Like the decomposition operator, the synthesis operator is usually used to improve the diversity of the population. The synthesis operator aims to transform two existing solutions $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ to new solution $\boldsymbol{\omega}'$ completely different from these two solutions. Here, we adopt the distance-preserving crossover operator [25] for the synthesis operator. First, all values that are equals and located on the same position in both parents will be copied to the offspring. The remaining positions of the offspring are randomly filled with the values not yet assigned. Note that, the value on the offspring is different from the values of the parents in the corresponding position.

before synthesis

$\boldsymbol{\omega}_1 : [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10]$

$\boldsymbol{\omega}_2 : [1,\ 4,\ 3,\ 2,\ 9,\ 6,\ 5,\ 8,\ 7,\ 10]$

$[1,\ \ ,\ 3,\ \ ,\ \ ,\ 6,\ \ ,\ 8,\ \ ,\ 10]$

after synthesis

$\boldsymbol{\omega}' : [1,\ 5,\ 3,\ 7,\ 2,\ 6,\ 9,\ 8,\ 4,\ 10]$

## 4.3 CVV

In this subsection we introduce the four operators of the CVV algorithm as well as the repair operator used to transform the unfeasible solutions. The CVV algorithm flowchart is shown in Fig. 3.

### 4.3.1 On-wall ineffective collision operator

In this operator, we adopt one physical machine change operator. More precisely, one virtual machine is randomly selected and placed on another physical machine. After that, the repair operator is used to ensure that the new generated solution is feasible. As shown below, the fourth virtual machine is taken from the physical machine 6 to the physical machine 2.

$[1,\ 1,\ 5,\ 6,\ 3,\ 4,\ 2,\ 5,\ 3,\ 4] \rightarrow [1,\ 1,\ 5,\ 2,\ 3,\ 4,\ 2,\ 5,\ 3,\ 4].$

### 4.3.2 Inter-molecular ineffective collision operator

The inter-molecular ineffective collision operator for the CVV algorithm is performed by: firstly, two different integers are randomly selected from the range [1,n], where n is the size of the solution. The new solutions should be repaired by the repair operator. Then, we exchange the values between the two randomly selected integers in the two solutions to produce two new different solutions. Finally, the repair operator is applied on the two new generated solutions. An example is given below. Besides, the values between positions 3 and 7 are exchanged.

before intermolecular ineffective collision

$\boldsymbol{\omega}_1 : [1,\ 1,\ \underline{5},\ 6,\ 3,\ 4,\ \underline{2},\ 5,\ 3,\ 4]$

$\boldsymbol{\omega}_2 : [1,\ 2,\ \underline{3},\ 4,\ 2,\ 3,\ \underline{6},\ 5,\ 4,\ 3]$

after intermolecular ineffective collision

$\boldsymbol{\omega}'_1 : [1,\ 1,\ \underline{5},\ 4,\ 2,\ 3,\ \underline{2},\ 5,\ 3,\ 4]$

$\boldsymbol{\omega}'_2 : [1,\ 2,\ \underline{3},\ 6,\ 3,\ 4,\ \underline{6},\ 5,\ 4,\ 3]$

### 4.3.3 Decomposition operator

In the decomposition operator, the values of parity position on the original solution are passed respectively to the two new solutions, and the values of the other position on the two new solutions are randomly generated from the range [1,m]. As for the operators mentioned above, the two new solutions should be repaired using the proposed repair operator. An illustrative example is given:

before decomposition

$\boldsymbol{\omega} : [1,\ 1,\ 5,\ 6,\ 3,\ 4,\ 2,\ 5,\ 3,\ 4]$

after decomposition

$\boldsymbol{\omega}'_1 : [1,\ 2,\ 5,\ 1,\ 3,\ 3,\ 2,\ 4,\ 3,\ 5]$

$\boldsymbol{\omega}'_2 : [2,\ 1,\ 1,\ 6,\ 2,\ 4,\ 3,\ 5,\ 5,\ 4]$

### 4.3.4 Synthesis operator

In the synthesis operator of the CVV algorithm, Each component of the new solution is equal to one of the two values located on the same position in two existing solutions with the same probability. The repair operator is also performed on the new solution to ensure its feasibility. An illustrative example is given below:

before synthesis

$\boldsymbol{\omega}_1 : [1,\ 1,\ 5,\ 6,\ 3,\ 4,\ 2,\ 5,\ 3,\ 4]$

$\boldsymbol{\omega}_2 : [1,\ 2,\ 3,\ 4,\ 2,\ 3,\ 6,\ 5,\ 4,\ 3]$

after synthesis

$\boldsymbol{\omega}' : [1,\ 2,\ 5,\ 6,\ 2,\ 3,\ 2,\ 5,\ 3,\ 4]$

### 4.3.5 Repair operator

During the generation of the new solutions by the different CVV operators, the constraints are not taken in consideration which may produce an infeasible solution. Hence, a repair operator is needed to check and transform the new generated solutions. In this paper, we design a new repair operator, it firstly selects the virtual machines that failed to be placed on their attributed PMs, this is because the corresponding physical machine does not have enough resources (CPU, RAM, or both of them). After that, we place those virtual machines using the following strategies: if the required resource CPU of the virtual machine is greater than the required resource RAM, then we place it on the physical machine that its available resources CPU is greater than RAM using the Best-Fit algorithm (BF), BF means that the virtual machine is placed into the physical machine which has enough resource and fewest resource remaining. If this placement is unfeasible, we place it on the PMs that having virtual machines inside using also the BF algorithm. If this placement is also impossible, we place it on the physical machine that does not contain any virtual machines and vice versa.

## 5 Simulation platform and results

In this section, we present and discuss various experimental tests to assess the performance of our proposed algorithms. In order to support the worst VMP placement scenario in which only one VM is assigned per server [40], The number of servers was set to the number of VMs. Moreover, for the sake of simplicity, the simulations of the homogeneous server environments are performed assuming that the offered resource is (1,1). In addition to the homogeneous server environments, our algorithm also can be used in case of heterogeneous servers. Note that, in order to verify the effectiveness of our algorithm, we do the simulations of the cloud computing platform as equivalent to the real large-scale cloud platform environment.

### 5.1 Calculation conditions

All the compared algorithms are coded with Java language and the test environment is set up on the same personal computer with processor Intel(R) i5-7500(R) 3.40 GHz, 16G RAM and Windows 10 Operating System.

**Table 2** Setting parameters of the two algorithm CVV and CVP

| Parameters | CVP | CVV |
|---|---|---|
| PopSize | 200 | 20 |
| CollRate | 0.7 | 0.1 |
| buffer | 0 | 0 |
| IniKE | Initial minimal fitness | Initial minimal fitness |
| LossRate | 0.2 | 0.1 |
| $\alpha$ | N | N*1000 |
| $\beta$ | 0.1*IniKE | 0.1*IniKE |

### 5.2 Parameters tuning and setting

A complete evaluation on all possible combinations of the parameters is impractical. Our goal is to assign parameter values to our algorithms with relatively good performance for the considered test instances. Parameter settings of the two algorithms are shown in Table 2 where N is the size of solution. The maximum number of function evaluations (FE) is set N*2000.

**Algorithm 1** Generation of VM machines

```
for i=1 to n do
  $vm_i^{ram}$=rand( $\overline{vm^{ram}}$);
  r=rand(1.0);
  if $(r < P \wedge vm_i^{cpu} \geq \overline{vm^{cpu}}) \vee (r \geq P \wedge vm_i^{cpu} < \overline{vm^{cpu}})$
    then $vm_i^{ram} = vm_i^{ram} + \overline{vm^{ram}}$
  end if
end for
```

We adopt the method used in [26] to randomly generate the problem instances. This method can generate random sequences of CPU and RAM utilization with several correlations. The main steps of this method are illustrated in Algorithm 1. Where rand(r) is a random function which can return an uniform distribution number in the range [0,r], $\overline{vm^{cpu}}$ and $\overline{vm^{ram}}$ are the reference utilization of CPU and RAM respectively, P is a probability values that are used to control the correlations of CPU and RAM utilization. For each instance, each test was repeated 20 times and the average results of these 20 independent runs are reported.

In the first and the second set of the experiments, we suppose that 200 VMs need to be placed and we set two kinds of reference utilization values for the CPU and RAM with five probabilities. Thus globally, we generate ten different scenario by setting $\overline{vm^{cpu}}$=$\overline{vm^{ram}}$=25% and $\overline{vm^{cpu}}$=$\overline{vm^{ram}}$=45% with five values (0.0, 0.25, 0.5, 0.75, 1.0). In the case of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ , the distributions of CPU and RAM utilization are in the range [0,50%) and [0,90%) respectively.

For $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ with five values, the average correlation coefficients are -0.754, -0.348, -0.072, 0.371, and 0.755. Those coefficients indicate the strong negative, weak negative, no, weak positive and the positive correlations for the five instances. Similarly, when we set five values for $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ , the coefficients are -0.755, -0.374, -0.052, 0.398 and 0.751 [27, 28].

### 5.3 Simulation results and discussion

The results of the proposed algorithms CVP and CVV are compared with four other heuristic algorithms which are FFD, BFD, CSO and RGGA algorithm [8, 29, 30, 38].

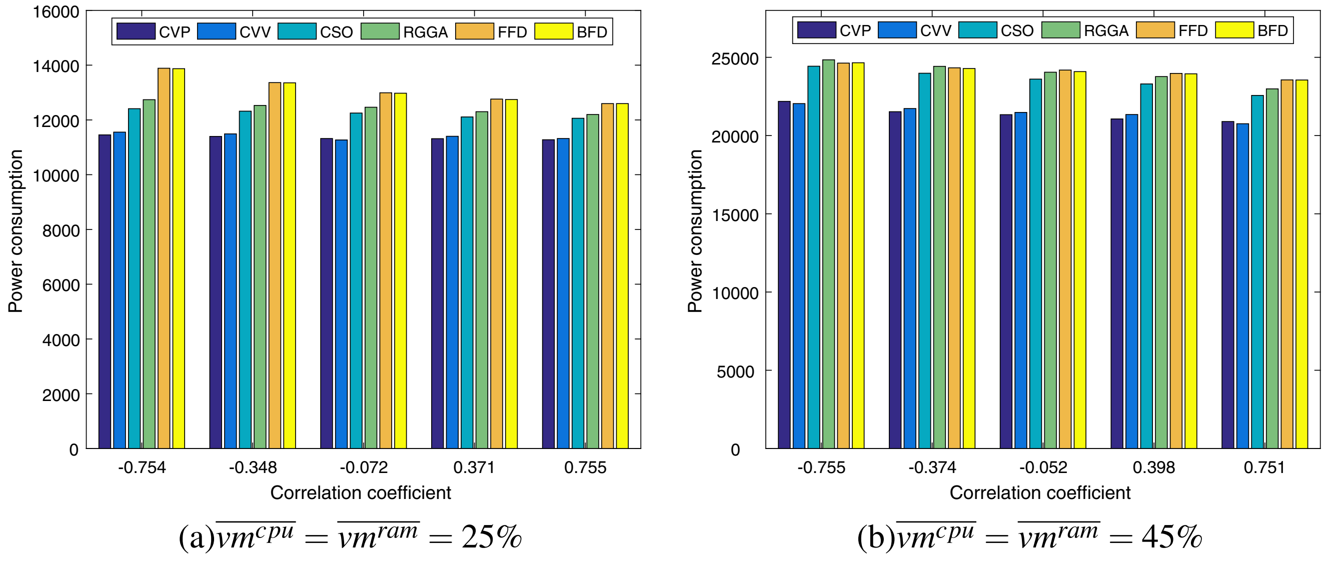


**Fig. 4** Power consumption of CVP, CVV, CSO, RGGA, FFD and BFD with 200 VMs for cases of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ (**a**) and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ (**b**)

On one hand, FFD considers the VMs in a decreasing order in terms of CPU utilization and places each VM into the first host that has enough resource remaining. Whereas, BFD puts the VMs in a decreasing order in term of CPU utilization and places each VM into the best host that has enough and fewest resource remaining. On the other hand, CSO implements a multi-objects algorithm to simultaneously optimize the power consumption with the resource utilization and RGGA employs a modified version of GGA's fitness function in order to solve the Multi-Capacity Bin Packing Problems.

Now, we are ready to present the experimental results of the proposed algorithms CVP and CVV with the other compared algorithms on solving the VMP. Two performanse metrics are used to assess the perfomrance of the compared algorithms: The average power consumption (W) and the average resource utilization.

In order to assess the energy consumption and evaluate the resources utilization of the different algorithms, Fig. 4 gives a complete comparison of the energy consumption for the algorithms CVP, CVV, CSO, RGGA, FFD and BFD with 200 VMs and Fig. 5 gives the differences of the resources

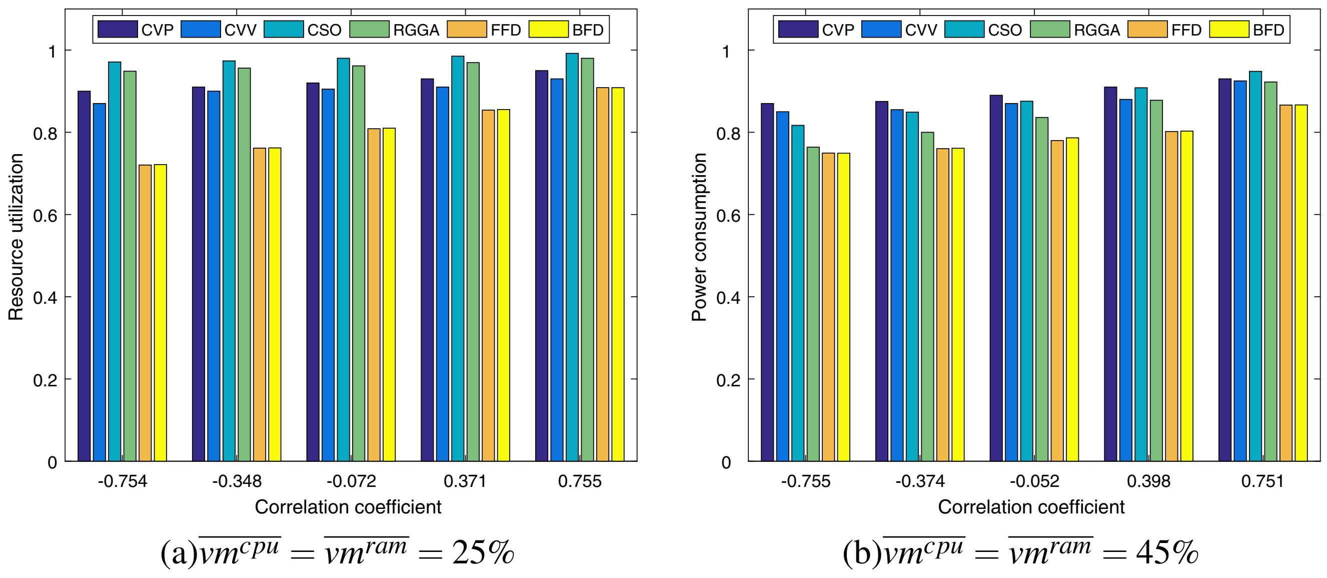


**Fig. 5** Resource utilization rate of CVP, CVV, CSO, RGGA, FFD and BFD with 200 VMs for cases of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ (**a**) and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ (**b**)

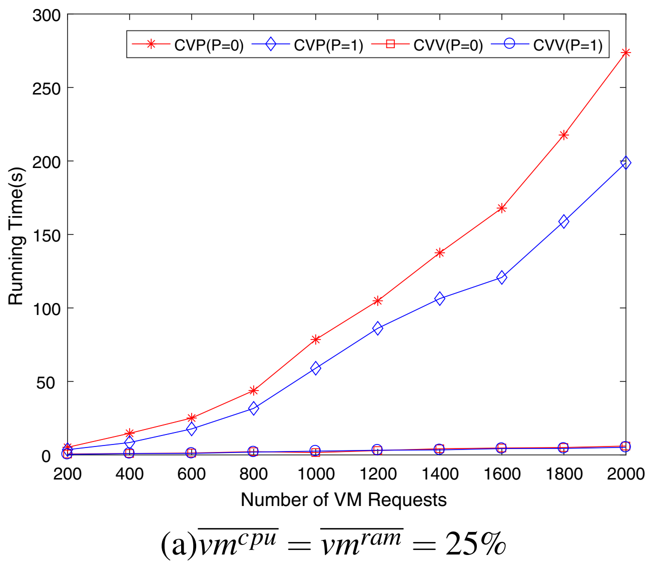
utilization rates of CVP, CVV, CSO, RGGA, FFD and BFD with 200 VMs.

Moreover, In order to evaluate the scalability performance of the proposed algorithms, the running time of two algorithms are depicted in Fig. 6 with a number of virtual machines varied from 200 to 2000.

From the experimental results of Figs. 4, 5 and 6, we can extract the following remarks:

1. In terms of the average energy consumption, in both cases where $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$, the CVP and CVV algorithms are better than other compared algorithms with an advantage to CVP algorithm comparing with other proposed algorithms. Note that, with the enhancement of correlation degree, the advantage gap between CVP and CVV is gradually reduced compared with other algorithms which indicates that CVP and CVV algorithms have better performance when CPU and RAM requirements are negatively correlated. As a result, we can deduce that the proposed CVP and CVV algorithm have a big advantage in terms of energy consumption. This advantage especially appears when CPU and RAM requirements are negatively correlated. Therefore, compared with other VMP algorithms, CVP and CVV have great advantages in terms of energy saving performance.
2. From Fig. 5, in case of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$, we can see that CVP and CVV algorithms have a slight disadvantage comparing with CSO and RGGA algorithms. However, CVP and CVV algorithms still outperform the others algorithms FFD and BFD. In case of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$, CVP and CVV start to show a better advantage comparing with the other compared algorithms. This advantage increases with the increase of the negative correlation coefficient. When CPU and RAM requirements are negatively correlated, we can remark that CVP and CVV algorithms have better effects comparing with CSO. However, when the correlation coefficients is close to 0.751, the CSO algorithm has the advantage.
3. In both cases where $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$. As the degree of correlation increases i.e.,from strong negative to strong positive, the average power consumption and average resource loss of all algorithms are decreased. This is because as the correlation increases, the number of servers required to place the VM decreases, resulting in lower power consumption and resource waste.

   However, when the correlation is strong and the correlation is strong, the energy consumption and resource utilization performance of the other four algorithms are very different, but CVP and CVV still show good performance in extreme cases. Therefore, we can conclude that, compared with CSO, RGGA, FFD and BFD, CVP and CVV are not very sensitive to correlation.
4. In order to test whether the two proposed algorithms are suitable for the large scale virtual machine placement. In the third experiments, we set $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ with $P = 0.0$ and $P = 1.0$. At the same time, the number of VMs from 200 to 2000. Figure 6 shows the variation of the execution time(s) with the number of virtual machines



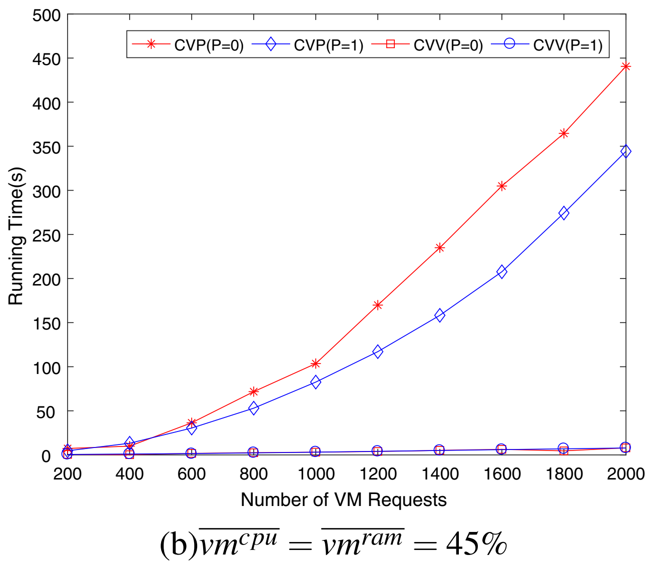


**Fig. 6** Running time of CVP and CVV with VMs from 200 to 2000 for case of $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ (**a**) and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ (**b**)

changed from 200 to 2000. We can draw the following conclusions:

a) In both cases where $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$ with $P = 0.0$ and $P = 1.0$, the execution time of CVV smoothly increases with an increase of the number of virtual machines. Whereas, the execution time of CVP increases rapidly with the increase of the number of virtual machines. However, the execution time of the CVP algorithm still accepted.
b) Compared $\overline{vm^{cpu}} = \overline{vm^{ram}} = 25\%$ and $\overline{vm^{cpu}} = \overline{vm^{ram}} = 45\%$, the former is less time-consuming. In terms of correlation, strong positive correlation is shorter than strong negative correlation.
c) Overall speaking, in terms of execution time, the CVP algorithm is more suitable for the small-scale virtual machines placement such as the level of racks or clusters. Whereas, the CVV has better scalability performance to deal with the large scale cloud computing platforms.

## 6 Conclusion and future work

Virtual machine placement has become one of the most popular research directions in cloud computing. In fact, how to place virtual machines on physical ones with the minimization of the energy consumption and the maximization of the resource utilization is an interesting problem to be solved. At the same time, when dealing with large-scale cloud computing platforms, not only the above two basic performances should be guaranteed, but also the time complexity of scheduling algorithm should be as low as possible in order to meet the quality of user service(QoS). In other words, users not only require the lowest possible cost but also need a fast user service response.

In this paper, we have proposed two algorithms namely CVP and CVV which are based on the basic CRO framework for solving virtual machine placement problem efficiently. Eight problem specific elementary reactions for the two algorithms are carefully designed to guarantee the local and global search capabilities. Moreover, a new repair operator is designed for CVV to repair the new unfeasible solutions after the four basic operators. From the experimental results, we have concluded that CVP and CVV have lower energy consumption and higher resource utilization comparing with other state of art algorithms which are CSO, RGGA, FFD and BFD. More precisely, CVV algorithm has shown an excellent performance in term of running time when dealing with large-scale cloud computing platforms.

For future works, our plan is to apply the two proposed virtual machine scheduling algorithms on the real cloud computing scheduling platform. Moreover, we plan to combine the domain knowledge with the proposed algorithms to guide the optimization process in order to reach better results.

**Acknowledgements** This work was partially supported by the National Natural Science Foundation of China (No.61672215, U1613209). Finally, we would like to thank Jiawei Qiu for his valuable comments and contributions to improving the quality of this article. The correspondence author is Zhiyong Li.

## References

1. Hwang I, Pedram M (2013) Hierarchical virtual machine consolidation in a cloud computing system. Cloud Comput 8201:196–203
2. Zheng X, Cai Y (2014) Dynamic virtual machine placement for cloud computing environments. In: International conference on parallel processing workshops, pp 121–128
3. Rimal BP, Choi E, Lumb I (2009) A taxonomy and survey of cloud computing systems. In: International joint conference on Inc, Ims and IDC, pp 44–51
4. Lee J (2013) A view of cloud computing. Commun Acm 53(4): 50–58
5. Mastroianni C, Meo M, Papuzzo G (2013) Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans Cloud Comput 1(2):1
6. Chang Y, Gu C, Luo F (2017) A novel energy-aware and resource efficient virtual resource allocation strategy in IaaS cloud. In: IEEE International conference on computer and communications, pp 1283–1288
7. Jain N, Choudhary S (2016) Overview of virtualization in cloud computing. Colossal Data Analysis and Networking
8. Sait SM, Bala A, El-Maleh AH (2016) Cuckoo search based resource optimization of datacenters. Appl Intell 44(3):489–506
9. Garg SK, Yeo CS, Buyya R (2011) Green cloud framework for improving carbon efficiency of clouds. Springer, Berlin, pp 491–502
10. Dasgupta G, Sharma A, Verma A, Neogi A, Kothari R (2011) Workload management for power efficiency in virtualized data centers. Commun Acm 54(7):131–141
11. Liu XF, Zhan ZH, Deng JD, Li Y, Gu T, Zhang J (2016) An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans Evol Comput PP(99):1–1
12. Bin E, Biran O, Boni O, Hadad E, Kolodner EK, Moatti Y, Lorenz DH (2011) Guaranteeing high availability goals for virtual machine placement. In: International conference on distributed computing systems, pp 700–709
13. Greenberg A, Hamilton J, Maltz DA, Patel P (2008) The cost of a cloud: research problems in data center networks. Acm Sigcomm Comput Commun Rev 39(1):68–73
14. Xiao Z, Qi C, Luo H (2014) Automatic scaling of internet applications for cloud computing services. IEEE Comput Soc, 1111–1123
15. Sahu Y, Pateriya RK, Gupta RK (2013) Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm. In: International conference on computational intelligence and communication networks, pp 527–531
16. Amokrane A, Zhani MF, Langar R, Boutaba R, Pujolle G (2013) Greenhead: virtual data center embedding across distributed infrastructures. IEEE Trans Cloud Comput 1(1):36–49
17. Sait SM, Shahid KhS (2015) Engineering simulated evolution for virtual machine assignment problem. Appl Intell 43(2):296–307

18. Lam AYS, Li VOK (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399
19. Truong TK, Li K, Xu Y (2013) Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem. Appl Soft Comput J 13(4):1774–1780
20. Xu J, Lam AYS, Li VOK (2011) Chemical reaction optimization for task scheduling in grid computing. IEEE Trans Parallel Distrib Syst 22(10):1624–1631
21. Nouioua M, Li Z (2017) Using differential evolution strategies in chemical reaction optimization for global numerical optimization. Appl Intell 4(1):1–27
22. Li JQ, Pan QK (2012) Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity. Appl Soft Comput 12(9):2896–2912
23. Duan H, Lu G (2015) Elitist chemical reaction optimization for contour-based target recognition in aerial images. IEEE Trans Geosci Remote Sens 53(5):2845–2859
24. Lam AYS, Li VOK, Yu JJQ (2012) Real-coded chemical reaction optimization. IEEE Trans Evol Comput 16(3):339–353
25. Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. ISCA, 13–23
26. Merz P, Freisleben B (1997) A genetic local search approach to the quadratic assignment problem. CiteSeer, 465–472
27. Ajiro Y, Tanaka A (2007) Improving packing algorithms for server consolidation. In: International computer measurement group conference, December 2–7, 2007. San Diego, Ca, USA, Proceedings, pp 399–406
28. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst 28(5):755–768
29. Shi L, Furlong J, Wang R (2013) Empirical evaluation of vector bin packing algorithms for energy efficient data centers. IEEE
30. Mustafa S et al (2016) Performance evaluation of energy-aware best fit decreasing algorithms for cloud environments. In: IEEE International conference on data science and data intensive systems, pp 464–469
31. Lpez-Pires F, Barn B, Benłtez L, Zalimben S, Amarilla A (2017) Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty. Futur Gener Comput Syst, 79
32. Zhang J et al (2013) Clustering based virtual machines placement in distributed cloud computing. Case-based reasoning research and development. Springer, Berlin, pp 233–240
33. Chen Y, Chen X, Liu W, Zhou Y, Zomaya AY, Ranjan R et al (2017) Stochastic scheduling for variation-aware virtual machine placement in a cloud computing cps. Future Generation Computer Systems
34. Sotiriadis S, Bessis N, Buyya R (2017) Self managed virtual machine scheduling in cloud systems. Information Sciences
35. Filho MC, Monteiro, Incio PR, Freire MM (2017) Approaches for optimizing virtual machine placement and migration in cloud environments: a survey. J Parallel Distrib Comput, 111
36. Abdel-Basset M, Abdle-Fatah L, Sangaiah AK (2018) An improved lvy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Comput (1), 1–16
37. Chen H (2016) A grouping genetic algorithm for virtual machine placement in cloud computing. In: International conference on collaborative computing: networking, applications and worksharing. Springer, Cham, pp 468–473
38. Wilcox D, Mcnabb A, Seppi K (2011) Solving virtual machine packing with a reordering grouping genetic algorithm. In: 2011 IEEE Congress on evolutionary computation (CEC). IEEE, pp 362C369
39. Luo J, Guo Y, Fu S, Li K, He W (2015) Virtual resource allocation based on link interference in cayley wireless data centers. IEEE Trans Comput 64(10):3016–3021
40. Chen S, Li Z, Yang B, Rudolph G (2016) Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems. IEEE Trans Parallel Distrib Syst 27(6):1796–1810
41. Shojafar M, Kardgar M, Hosseinabadi AAR, Shamshirband S, Abraham A (2015) TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan. AMS
42. Gupta MK, Jain A, Amgoth T (2018) Power and resource-aware virtual machine placement for IaaS cloud. Sustainable Computing: Informatics and Systems

**Zhiyong Li** received the MSc degree in System Engineering from National University of Defense Technology, Changsha, China, in 1996 and PhD degree in Control Theory and Control Engineering from Hunan University, Changsha, China, in 2004.

Since 2004, he joined the College of Computer Science and Electronic Engineering of Hunan University. Now, he is a Full Professor with Hunan University, member of IEEE, China Computer Federation (CCF) and Chinese Association for Artificial Intelligence (CAAI). His research interests include Intelligent Perception and Autonomous Moving Body, Machine Learning and Industrial Big Data, Intelligent Optimization Algorithms with Applications. He has published more than 100 papers in international journals and conferences.

**Yang Li** graduated from Anhui university of engineering with a B.S. degree in information and computing science in 2016. Currently he is studying for a master's degree in computer science and technology at College of Computer Science and Electronic Engineering Hunan University, Changsha, China. His research interests including evolutionary computation, virtual machine scheduling optimization and computer vision.

**Tingkun Yuan** received the MSc degree in Software engineering from College of Computer Science and Electronic Engineering Hunan University, Changsha, China, in 2017. His research interests including evolutionary computation, and scheduling optimization.

**Shaomiao Chen** received the PhD degree in computer science and technology from College of Computer Science and Electronic Engineering Hunan University, Changsha, China, in 2018. Currently, He is a teacher of Hunan University of Science and Technology. His research interests include parallel computing, evolutionary computation, and scheduling optimization.

**Shilong Jiang** Jiang received the B.S. degree from the Department of Automation Control, Taiyuan Institute of Mechanical, Taiyuan, PRC, in 1991, the MSc degree from the School of Information Engineering, Central South University of Technology, Changsha, PRC, in 1994, and the Ph.D. degree from Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, in 2000. He is with director in Laboratory of Motion Control Technology, PKU-HKUST Shenzhen-HongKong Institution. His research interests include intelligent robotics, machine vision, IoT, VR/AR, sensor fusion, multifingered manipulation, and motion control.