Hindawi
Scientific Programming
Volume 2020, Article ID 8846087, 8 pages
https://doi.org/10.1155/2020/8846087

Research Article

Location-Constrained Virtual Machine Placement (LCVP) Algorithm

Zhenpeng Liu,[1,2] **Jiahuan Lu**,[1] **Nan Su**,[1] **Bin Zhang**,[2] **and Xiaofei Li**[2]

[1]*School of Cyber Security and Computer, Hebei University, Baoding, Hebei 071002, China*
[2]*Information Technology Center, Hebei University, Baoding, Hebei 071002, China*

Correspondence should be addressed to Xiaofei Li; lixiaofei@hbu.edu.cn

Received 24 September 2020; Revised 16 October 2020; Accepted 19 October 2020; Published 6 November 2020

Academic Editor: Cristian Mateos

Virtual machine (VM) placement is the current day research topic in cloud computing area. In order to solve the problem of imposing location constraints on VMs to meet their requirements in the process of VM placement, the location-constrained VM placement (LCVP) algorithm is proposed in this paper. In LCVP, each VM can only be placed onto one of the specified candidate physical machines (PMs) with enough computing resources and there must be sufficient bandwidth between the selected PMs to meet the communication requirement of the corresponding VMs. Simulation results show that LCVP is feasible and outperforms other benchmark algorithms in terms of computation time and blocking probability.

1. Introduction

The evolutionary advancements in the field of technology have led to the instigation of cloud computing [1]. With the popularity of cloud computing, VM placement has received more and more attention. Jobs arrive from the cloud consumer and are analyzed to produce the multiple corresponding subtasks that VM can run directly. The physical resources provided by a data center are used to build VMs to support the execution of these subtasks. The part playing a connecting role between the two mentioned above is called VM placement, which is concerned with mapping VMs to PMs in a data center [2, 3].

There has been some research in VM placement. The traditional approaches consider VM placement as a well-known bin-packing problem [4, 5], which assumes that the performance of task execution can always satisfy cloud consumers as long as the requested computing resources are less than the total available resources in a data center. However, this assumption ignores factors such as available bandwidth resources and geographical distances between the selected PMs. Specifically, if available physical resources in a data center are sufficient but belong to the different PMs far away from each other, using these resources to build VMs may lead to network congestion easily, which further detain the task execution [6]. Based on this consideration, the authors in [6] proposed a new algorithm for data-intensive distribution applications, which can promote the effective execution of these applications by giving bandwidth higher priority over PMs. Unfortunately, the performance of the algorithm cannot be guaranteed and it is also possible to end up with worse solutions. In order to finish the task on time, a backfilling algorithm to execute deadline-based tasks was proposed [7]. By introducing the idea of the double auction, the mechanism in [8] can bridge users' task requirements and providers' resources in two-side cloud markets and achieve the purpose that the purchase prices of physical resources for building VMs are as close as possible to their true value. Mann et al. [2] found that VM placement and VM selection influence each other significantly and in a highly nontrivial way. According to this, they proposed a problem formulation for the joint optimization of them. However, they only gave some primary theories and there is still much work for further research. Based on [2], Pascual et al. proposed a new combined optimization model to study how a task affects others on the same PM [9], according to the sizes and types of user tasks. The authors in [10] dug into the design and implementation of virtual machine management

strategies for energy-efficient cloud data centers and proposed a distributed approach to an energy-efficient dynamic virtual machine consolidation mechanism. In order to enhance security, a virtual resource mapping algorithm was proposed in [11], which can configure resources based on evaluating and detecting the threats and vulnerabilities of VMs. However, the algorithm missed the opportunity to place multiple VMs onto one PM, which reduces the resource utilization of a data center. Zhao et al. proposed a function model between the performance of task executions, the resource costs, and the impact of multiple tasks on the same PM [12]. And then they proposed an optimization algorithm based on the model. Cortez et al. used machine learning to predict the resource costs of VMs to achieve the purposes of reserving resources and improving resource utilization more effectively [13].

In cloud computing, the data center acts as an infrastructure to provide physical resources for VMs [14] and one of the important problems is determining mappings of VMs to PMs with different objectives, such as optimizing costs, profits, or performances [15]. As the basic unit to supply services for cloud consumers, VMs will be placed very frequently. If multiple VMs that need to communicate with each other are placed onto the PMs far away from each other, substantial bandwidth will be occupied inevitably to maintain communication between the VMs, which will cause unnecessary waste of network resources and may lead to network congestion [11]. Moreover, there are usually thousands of PMs in a data center. If the PMs are searched for numerous cloud user requests without any constraints, the complexity of resource management will increase exponentially. Therefore, considering the factors mentioned above and the requirements of functionality, security, availability [16], it is necessary to impose constraints on VMs and the constraints come from location requirements in this article.

In summary, the major contributions of this paper are as follows:

(i) We provide a new perspective for VM placement and formulate the problem of imposing location constraints on VMs in cloud computing

(ii) Based on this new perspective, we propose an algorithm to generate the desired solution

(iii) We conduct and analyze extensive simulations to demonstrate the effectiveness of LCVP. The results show that our algorithm achieves better performance and lower computation time

In this paper, the LCVP algorithm that can consider the location constraints is proposed. Simulation results show that, compared with the existing ones, LCVP can achieve its goal and outperform other benchmarks in terms of computation time and blocking probability.

The rest of this paper is organized as follows. In Section 2, the problem description illustrates the problem solved in this paper first. And then the models are presented in model definition. The feasible solution is formulated in the rest of Section 2. Section 3 describes the LCVP algorithm and its performance evaluation is done in Section 4. Finally, the conclusion is given in Section 5.

2. VM Placement

2.1. Problem Description. As mentioned above, it is necessary to impose location constraints on VMs and the problem solved in this paper is mapping the requested VMs onto the appropriate PMs under multiple constraints. Specifically, each VM should be placed onto one of the specified candidate PMs that are defined based on the preferred location and radius, i.e., location constraint. Each selected candidate PM should have sufficient computing resources to host the corresponding VM, i.e., computing capacity constraint. And there should be sufficient bandwidth between each pair of the selected PMs to maintain the communication between the corresponding VMs, i.e., bandwidth capacity constraint. The objective of LCVP is to serve as many customer requests as possible and maximize the resource utilization of a data center based on multiple constraints.

2.2. Model Definition. The physical resources provided by a data center to supply cloud computing services are represented as an undirected graph called PM-graph. The VMs needed by a cloud consumer to run his/her tasks are represented as an undirected graph called VM-graph.

2.2.1. PM-Graph. It indicates the physical resources provided by a data center. The PM-graph is defined as an undirected graph $G^s(V^s, E^s)$, where V^s represents the set of PMs and E^s represents the set of communication between PMs. The communication is called the physical link in the following sections. Each PM $v^s \in V^s$ has a computing capacity $c^s_{v^s}$ and each physical link $e^s \in E^s$ has a bandwidth capacity$b^s_{e^s}$. In addition, each PM is also associated with a location $l^s_{v^s}$. P^s is the set of loopless paths in $G^s(V^s, E^s)$ and $P^s_{v^s}$ is the set of loopless paths that start/end at node v^s.

2.2.2. VM-Graph. It indicates a customer request, or in other words, the VMs needed by a cloud customer to finish his/her jobs before the deadline. The VM-graph is defined as an undirected graph $G^r(V^r, E^r)$, where V^r is the set of VMs requested by a cloud consumer and E^r is the set of communication that represents bandwidth requirements to support data flow between the corresponding VMs. The communication between VMs is called virtual link (VL) in the following sections. Each VM $v^r \in V^r$ has a computing requirement $c^r_{v^r}$ and each VL $e^r \in E^r$ has a bandwidth requirement $b^r_{e^r}$. In addition, for LCVP, each VM $v^r \in V^r$ has a preferred location, denoted as $l^r_{v^r}$. Based on $l^r_{v^r}$, VM $v^r \in V^r$ can only be placed onto the candidate PM(s) that is(are) defined based on the preferred location $l^r_{v^r}$ and a radius ρ, i.e., location constraints mentioned in this paper. The set of candidate PM(s) is denoted as $\Phi^s_{v^r}$, i.e.,

$$\begin{aligned} &\Phi^s_{v^r} \subseteq V^s, \\ &\Phi^s_{v^r} = \left\{ u^s \in V^s : \left\| l^s_{u^s} - l^r_{v^r} \right\| \le \rho \right\}, \end{aligned} \tag{1}$$

where $||l^s_{u^s} - l^r_{v^r}||$ is the distance between the two locations.

Each VL $e^r \in E^r$ is also associated with a candidate physical path set $P^s_{e^r}$, which includes all the paths between the candidate PMs of the two end-nodes of e^r, i.e.,

$$P^s_{e^r} = \bigcup_{v^s_1 \in \Phi^s_{e^r_+}} \bigcup_{v^s_2 \in \Phi^s_{e^r_-}} P^s_{v^s_1, v^s_2}, \tag{2}$$

where e^r_+ and e^r_- are the two end-nodes of e^r. $\widehat{P}^s_{e^r}$ indicates the subset of $P^s_{e^r}$ with sufficient bandwidth to carry VL e^r.

VM-graphs are derived from the customer requests. Specifically, a cloud consumer submits jobs and then the jobs are analyzed to generate the corresponding subtasks that can be run directly on VMs. After this, according to the degree of parallelism in subtasks and the communication dependencies among those subtasks, the corresponding VM-graph is generated.

2.2.3. Compatibility Graph. The CG is a graph structure. It is denoted as $G^c(V^c, E^c)$, in which each node $v^c \in V^c$ represents a candidate physical path for a VL $e^r \in E^r$, and the nodes in the same row represent all candidate physical paths for the same VL, i.e.,

$$f_C(e^s_1) = f_C(e^s_2), \quad \text{if } f^{-1}_N(e^s_1) = f^{-1}_N(e^s_2), \tag{3}$$

where $f_C(\,)$ is the function to obtain the line number of the corresponding nodes in a CG and $f^{-1}_N(\,)$ is the inverse function of $f_N(\,)$ to obtain the corresponding virtual link that e^s_1 is a candidate for.

Each row of the CG represents all candidate physical paths for a particular VL in the VM-graph, i.e.,

$$P^s_{e^r} = \{e^s \in E^s : f_C(e^s_1) = f_C(e^s_2)\}. \tag{4}$$

Each link in a CG denotes the corresponding end-nodes are compatible. Specifically, if two physical paths are compatible, a link is inserted to connect their corresponding nodes in CG. Similarly, there is no link between the incompatible nodes.

Here, "compatible" means that two compatible physical paths can carry two VLs in the same VM-graph simultaneously. Specifically, the compatible physical paths should satisfy the following: (1) they are the candidate paths for two adjacent VLs and have one PM as the common end-node or (2) they are the candidate paths for two VLs that are not adjacent.

2.3. Feasible Solution. A feasible solution should satisfy the following demands. In a feasible solution, each selected PM to host the corresponding VM should satisfy the location constraint, computing capacity constraint, and one-to-one mapping constraint. Each selected physical link for the corresponding VL should satisfy the bandwidth capacity constraint and link mapping constraint. On this basis, as many VMs as possible should be placed to maximize the resource utilization of a data center. Specifically, each VM can only be placed onto the candidate PMs, i.e.,

$$v^s \in \Phi^s_{v^r}, \quad \text{if } f_N(v^r) = v^s, \quad \forall v^r \in V^r, \tag{5}$$

where $f_N(\,)$ denotes the mapping relation between VMs and its candidates.

Each VM in the same VM-graph can only be placed onto a single PM and any two different VMs in a single VM-graph cannot be placed onto the same PM, i.e.,

$$f_N(v^r_1) = f_N(v^r_2), \quad \text{if and only if } v^r_1 = v^r_2. \tag{6}$$

If a VM is split and placed onto multiple PMs, additional bandwidth will be occupied inevitably to support the VM's internal communication. And if multiple VMs are placed onto one PM, the customer request will be vulnerable to physical resources failures [17]. In addition, multiple VMs sharing the same PM are vulnerable to resource competition, which may cause performance interference among VMs and thus lead to VM performance degradation [18]. Each selected PM should have sufficient resources for the corresponding VM, i.e.,

$$c^r_{v^r} \le c^s_{v^s}, \quad \text{if } f_N(v^r) = v^s, \quad \forall v^r \in V^r. \tag{7}$$

Furthermore, each VL should be placed onto one physical path connecting the PMs that its two end-nodes are placed onto, i.e.,

$$f_L(e^r) \subseteq P^s_{f_N(e^r_+), f_N(e^r_-)}, \quad \forall v^r \in V^r, \tag{8}$$

where $f_L(\,)$ is the link mapping relation between VLs and physical paths and e^r_+ and e^r_- are the two end-nodes of VLe^r.

The allocated bandwidth on each physical link should not exceed its bandwidth capacity, i.e.,

$$\begin{aligned} b^s_{e^s} &\ge \sum_{e^r \in E^r} \sum_{P^s \in f_L(e^r)} b^{e^r}_{p^s} \cdot I^{e^s}_{p^s}, \forall e^s \in E^s, \\ b^r_{e^r} &= \sum_{p^s \in f_L(e^r)} b^{e^r}_{p^s}, \forall e^r \in E^r, \end{aligned} \tag{9}$$

where $b^{e^r}_{p^s}$ indicates the bandwidth allocated to accommodate VL e^r on the physical path p^s and $I^{e^s}_{p^s}$ indicates whether p^s traverses e^s or not, i.e.,

$$\begin{cases} I^{e^s}_{p^s} = 1, & \text{if } p^s \text{ traverses } e^s, \\ I^{e^s}_{p^s} = 0, & \text{otherwise.} \end{cases} \tag{10}$$

It should be clearly noted that there may be one or more feasible solutions, but only one of them will be the desired one that LCVP finally provides to the corresponding cloud consumer. As illustrated in the previous section, the objective of LCVP is to serve as many customer requests as possible and maximize the resource utilization of a data center. Hence, LCVP will select one with lower blocking probability from feasible solutions for a cloud consumer, that is, the desired solution.

3. LCVP Algorithm

3.1. Preprocessing. First of all, LCVP will preprocess all candidate PM sets upon receipt of the VM-graph and PM-graph. This is because of the one-to-one mapping between VMs and

PMs, that is, $f_N(v_1^r) = f_N(v_2^r)$ if and only if $v_1^r = v_2^r$, as previously mentioned. Through the preprocessing, LCVP can guarantee that each PM only is one candidate for one VM, that is, $\Phi^s_{v_1^r} \cap \Phi^s_{v_2^r} = \varnothing, \{v_1^r, v_2^r : v_1^r \neq v_2^r\}$. The detailed procedure is given in Algorithm 1.

3.2. CG Construction. After preprocessing, LCVP constructs CG next by using the VM-graph and PM-graph. The CG structure has been mentioned in the previous section, so it will not be repeated in this section. The body of CG construction is shown in Algorithm 2.

3.3. Heuristic Maximum Clique Algorithm. With CG, the problem of imposing location constraints on VM is transformed into the maximum clique problem. Since the nodes from the same row of CG represent all candidate physical paths for a particular VL, a node selected from a row in the CG means a VL has found a feasible physical link. The feasible solution has been found when LCVP found one node from each row of the CG. So, the next thing that needed to be done is to find the maximum clique that can minimize the total resource consumption from the constructed CG, that is, the desired solution. The resource consumption is calculated as follows:

$$R = \sum_{v^r \in V^r} c^r_{v^r} + \sum_{e^r \in E^r} \sum_{p^s \in f_L(e^r)} \left|p^s\right| * b^{e^r}_{p^s} \leq Q, \tag{11}$$

where f_L is the function between the selected physical link and virtual link e^r and $b^{e^r}_{p^s}$ is the bandwidth allocated on p^s for e^r in PM-graph.

The CG has a good property that a maximal clique in it is also the maximum one as long as there are sufficient bandwidth resources, which has been proven in [19]. Specifically, the feasible solution exists as long as there is at least one node that can be selected in each row. With this property, the problem of finding the maximum clique is reduced to finding a maximal one, which can optimize the computation time of LCVP.

Considering the purpose of load-balancing, the weight h of the candidate physical path is defined as its hop-count divided by its available bandwidth. According to the cannikin law, the available bandwidth of a physical path depends on the physical link with the smallest available capacity:

$$h = \frac{\left|p^s\right|}{\delta + \min_{e^s \in p^s} b^s_{e^s}}, \quad \forall p^s \in \widehat{P}^s_{e^r}, \tag{12}$$

where $|p^s|$ denotes the hop-count of physical path carrying virtual link e^r and δ is a small positive number to avoid zero denominators.

As mentioned previously, LCVP will find the desired maximal clique from the constructed CG by using the heuristic maximum clique algorithm shown below. f_{SP} is the function to obtain the physical path that a node in the CG represents, while f^{-1}_{SP} is the inverse function (Algorithm 3).

3.4. Desired Solution Generation. In the end, since each row of a CG represents all the candidate physical paths of a particular VL and each node represents a candidate physical path for a VL, the desired solution can be generated by traversing the maximum clique found in the previous section.

4. Results and Discussion

4.1. Performance Metrics. The objective of LCVP is to serve as many customer requests as possible and maximize the resource utilization of a data center. The solution found through LCVP should satisfy the constraints (5)–(9), that is, feasible solution. Hence, the customer request blocking probability is used as the performance metrics.

The so-called blocked request means that, due to insufficient physical resources in a data center, LCVP fails to generate the feasible solution.

Blocking Probability: it is defined as the ratio of blocked to total arrived requests, i.e.,

$$P_b = \lim_{T \longrightarrow \infty} \frac{\left|\Omega_b(T)\right|}{\left|\Omega_a(T)\right| + \left|\Omega_b(T)\right|}, \tag{13}$$

where $\Omega_a(T)$ and $\Omega_b(T)$ denote the set of the accepted and blocked customer requests during [0,T], respectively.

4.2. Simulation Step. In order to verify the feasibility and time-efficiency of LCVP, simulation experiments are conducted. The simulation environment is MATLAB 2012b running on a computer with 3.10 GHz Intel Core i3-2100 CPU and 4.00 GB RAM. The PM-graph and all VM-graphs are randomly generated with the GT-ITM [20], which is a tool for randomly generating network topology. There are 50 PMs and 172 physical links in the PM-graph and all the PMs are located within a 100×100 grid.

The PM-graphs used in the simulations are shown in Figures 1 and 2.

In each VM-graph, the number of VMs is between 2 and 10 randomly and the probability of connecting any two VMs is 0.5. The preferred location $l^r_{v^r}$ of each VM is also randomly located in the 100×100 grid and the candidate PM set for each VM consists of the PMs located within the circle that is centered at $l^r_{v^r}$ and has a radius of ρ. The default value of ρ is 20. The VM-graphs mentioned above are generated according to the Poisson process. The average arrival rate of the Poisson process is λ VM-graphs per time-unit and the average holding time of each VM-graph is $1/\mu$ time-units. Therefore, the traffic load of VM-graphs is λ/μ in Erlangs. To facilitate the following experiments, Yen's algorithm [21] is used to precalculate K-shortest physical paths between each PM-pair.

The algorithms proposed in [22] and their modified versions are used as the benchmarks, which are denoted as DViNE, DViNE-LB, DViNE-KSP, and DViNE-LB-KSP, respectively. The algorithms ending with "KSP" are the modified versions that precalculate K shortest physical paths compared with the original algorithms, while those including "LB" are the benchmark algorithms considering load

```
input : Original candidate PM sets Φ^S_{v^r}, PM set V^s, VM set V^r
output : Preprocessed candidate PM sets Φ^S_{v^r}
(1)  Ψ_{v^s} ← ∅, ∀v^s ∈ V^s;
     //check the candidate PM set for each VM and remove the PMs with insufficient capacity.
(2)  for each v^r ∈ V^r do
(3)    for each v^s ∈ Φ^s_{v^r} do
(4)      if c^r_{v^r} > c^s_{v^s} then
(5)      remove v^s from Φ^s_{v^r} ;
(6)    endif
(7)   endfor
(8)  endfor
     //for each PM, find out all VMs that consider the PM as a candidate and save the VMs in Ψ_{v^s}.
(9)  for each v^r ∈ V^r do
(10) for each v^s ∈ Φ^s_{v^r} do
(11)   Ψ_{v^s} ← Ψ_{v^s} ∪ {v^r};
(12) endfor
(13) endfor
     //for each PM whose Ψ_{v^s} > 1, only reserve it as the candidate for the VM whose candidate PM set
     //has the smallest size and remove it from other candidate sets.
(14) for each v^s ∈ V^s do
(15)  if |Ψ_{v^s}| > 1 then
(16)   v^r_m = arg min_{v^r ∈ Ψ_{v^s}} |Φ^s_{v^r}|;
(17)   Ψ_{v^s} ← Ψ_{v^s} \ {v^r_m};
(18)   for each v^r ∈ Ψ_{v^s} do
(19)    Φ^s_{v^r} ← Φ^s_{v^r} \ {v^s};
(20)   endfor
(21)  endif
(22) endfor
```

Algorithm 1: Preprocessing.

balance. In addition to the algorithms ending with "KSP," the others are the original ones proposed in [22]. LCVP also precalculates the K shortest physical paths by using Yen's algorithm and uses these paths to construct CG. Hence, there is no modified version ending with "KSP" for LCVP. The reason for using the modified versions is that the original ones did not apply limitation on the number of candidate physical paths for each virtual link, which is impractical for the data center with massive resources.

4.3. Simulation Results. On the PM-graph shown in Figures 1 and 2, the simulation experiment was conducted for 500000 time-units under a fixed traffic load as 20 Erlangs to evaluate the performance of the algorithms. The results are shown in Figure 3.

From Figure 3, it can be seen that the blocking probability of LCVP maintains about 6%, which is lower than the benchmark algorithms, thanks to the fact that LCVP can simultaneously consider the capacities of PMs, bandwidth capacities, and hop-count of physical links. Moreover, the algorithms considering the network bandwidth load have a lower probability than those that do not. It is obvious that the algorithms considering the network bandwidth load prefer to conserve sufficient bandwidth for each physical link to achieve load balance, which can provide more available physical links for subsequent customer requests.

The experiments are also conducted to compare the computation times between LCVP and benchmark algorithms. In order to show the results more directly and briefly, the computation time of LCVP is used as the basis and the results on the normalized computation time are shown in Figure 4. The algorithms that precalculate the K shortest physical paths are faster than the others because they not only save the time to find the paths but also prevent the algorithms from the unlimited search for potential candidate paths, which leads to a longer search time. In addition, it should be emphasized that the computation time of LCVP includes the time of using Yen's algorithm, preprocessing, constructing CG, and finding the maximum clique. It can be seen that even when the radius ρ is set as 40, the computation time of LCVP is still much lower than the benchmarks. This is due to the good property of CG, which optimizes the complexity of LCVP.

Finally, the relationship between the blocking rate and radius ρ is experimented and analyzed. For different values of radius ρ, i.e., different sizes of candidates PM sets, the results are shown in Figure 5. It can be seen that even when the radius ρ is 15, the blocking probability of LCVP is only 7%, which is much lower than the benchmarks. Furthermore, with the increase of radius ρ, the probability decreased to 1%. This is because when radius ρ increases, the number of candidate PMs for each VM also increases, which provides more choices for LCVP to find the desired solution. However, benchmark algorithms do not change significantly with the increase of radius ρ, whose reason is that the algorithms have no performance guarantee and may generate worse or even infeasible solutions.

input : PM-graph G^s , VM-graph G^r
output : CG $G^c(V^c, E^c)$
(1) $V^c \leftarrow \varnothing, E^c \leftarrow \varnothing$;
//check the candidate physical paths set $P^s_{e^r}$ for each VL and remove the paths with insufficient
//capacity.
(2) for each $e^r \in E^r$ do
(3) for each $p \in P^s_{e^r}$ do
(4) if $\min_{e^s \in p} b^s_{e^s} \geq b^r_{e^r}$ then
(5) Insert a node in V^c to represent p;
(6) else
(7) remove p from $P^s_{e^r}$;
(8) endif
(9) endfor
(10) if $P^s_{e^r} = \varnothing$ then
(11) return (FALSE);//return construction failure status
(12) endif
(13) endfor
//according to compatible relation between physical paths, connect the corresponding nodes in the
//CG.
(14) for each $e^r_1 \in E^r$ do
(15) for each $e^r_2 \in E^r, e^r_2 \neq e^r_1$ do
(16) if e^r_1 and e^r_2 are adjacent through VN v^r then
(17) for each $v^s \in \Phi^s_{v^r}$ do
(18) $E^c \leftarrow E^c \cup ((P^s_{e^r_1} \cup P^s_{v^s}) \times t(P^s_{e^r_2} \cup P^s_{v^s}))$;
(19) endfor
(20) else
(21) $E^c \leftarrow E^c \cup (P^s_{e^r_1} \times P^s_{e^r_2})$;
(22) endif
(23) endfor
(24) endfor
(25) return (TRUE);//return construction success status

Algorithm 2: CG construction.

input : CG G^c, PM-graph G^s, VM-graph G^r
output : Maximum clique M
(1) $M \leftarrow \varnothing$;
(2) $P^N \leftarrow f_{SP}(V^c)$;//put all corresponding physical links of the nodes in CG into the set P^N
(3) for each $e^r \in E^r$ in nonincreasing order of $b^r_{e^r}$ do
//obtain the physical links with sufficient bandwidth to carry e^r and store them in $\widehat{P}^s_{e^r}$
(4) $\widehat{P}^s_{e^r} \leftarrow \left\{p^s : p^s \in p^s_{e^r} \cap P^N, (\min_{e^s \in p^s} b^s_{e^s}) \geq b^r_{e^r}\right\}$;
(5) if $\widehat{P}^s_{e^r} = \varnothing$ then
(6) return (FALSE); //return failure status
(7) endif
select p^s_m from $\widehat{P}^s_{e^r}$ with the minimum h-parameter defined by equation (12);
(8) $M \leftarrow M \cup \left\{f^{-1}_{SP}(p^s_m)\right\}$;
(9) $P^N \leftarrow P^N \cap \left\{p^s : (f^{-1}_{SP}(p^s), f^{-1}_{SP}(p^s_m) \in E^c)\right\}$;//only conserve the compatible physical links as candidates for following virtual links.
(10) update computing resources;
(11) endfor
(12) update computing resources;
(14) return (TRUE); //return success status

Algorithm 3: Heuristic maximum clique algorithm.

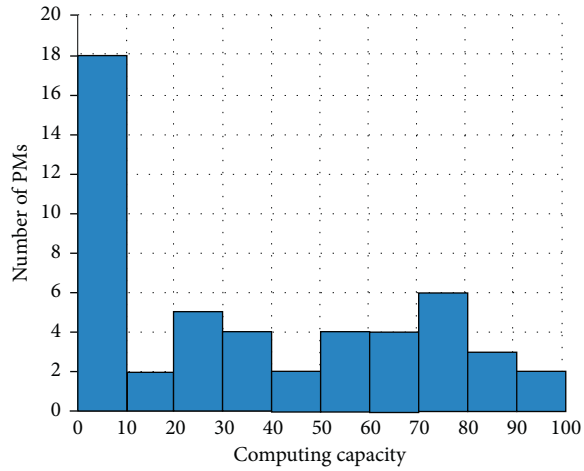

Figure 1: Computing capacity distributions used in the simulations.

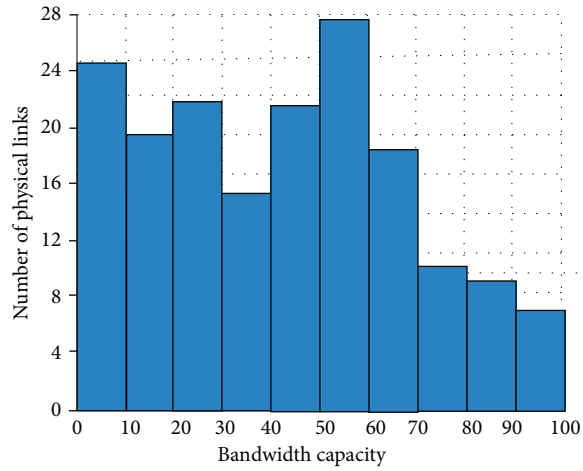

Figure 2: Bandwidth capacity distributions used in the simulations.

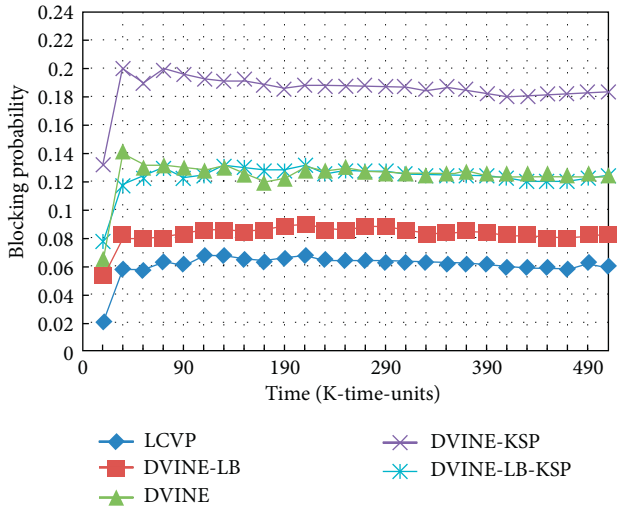

Figure 3: Simulation results on blocking probability.

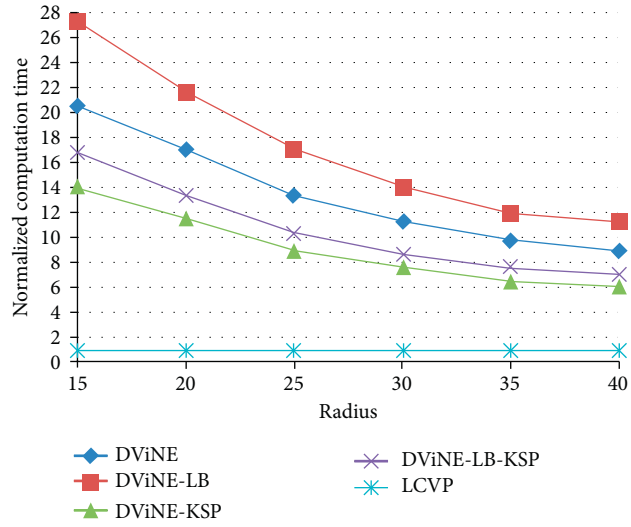

Figure 4: Normalized computation time versus radius.

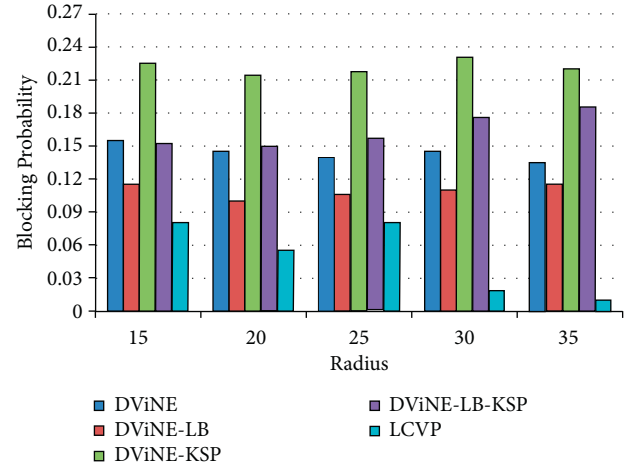

Figure 5: Blocking probability versus radius.

5. Conclusions

In cloud computing, the VM placement is an important research direction. The LCVP proposed in this paper can place VMs onto PMs under location constraints and ensure that there is at least one physical link between the selected PMs. The experimental results showed that LCVP could serve more customer requests and provide better blocking probability than the benchmarks with much lower computation time. In the future, the focus will be on the modified version of LCVP that can consider multipath based link mapping.

Data Availability

The data used can be found at https://pan.baidu.com/s/1-bjl1P6TM_qJQB09ctfcXw.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Conceptualization was performed by Zhenpeng Liu and Xiaofei Li. Methodology was prepared by Bin Zhang. Software was provided by Jiahuan Lu. Validation was carried out by Jiahuan Lu and Nan Su. Formal analysis, visualization, and supervision were conducted by Xiaofei Li. Investigation, project administration, and funding acquisition were performed by Zhenpeng Liu. Resources were provided by Bin Zhang and Jiahuan Lu. Data curation was conducted by Bin Zhang. Original draft was prepared by Zhenpeng Liu, Xiaofei Li, and Jiahuan Lu. Reviewing and editing were carried out by Jiahuan Lu. All the authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was funded by the Natural Science Foundation of Hebei Province (no. F2019201427) and the Ministry of Education, "Cloud Integration, Science and Education Innovation" Fund Project of China (no. 2017A20004).

References

[1] J. P. Martin, A. Kandasamy, K. Chandrasekaran, and C. T. Joseph, "Elucidating the challenges for the praxis of fog computing: an aspect-based study," *International Journal of Communication Systems*, vol. 32, no. 7, pp. e3926–e392628, 2019.

[2] Z. A. Mann, "Resource optimization across the cloud stack," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 1, pp. 169–182, 2018.

[3] N. D. Vahed, M. Ghobaei-Arani, and A. Souri, "Multi-objective virtual machine placement mechanisms using nature inspired metaheuristic algorithms in cloud environments: a comprehensive review," *International Journal of Communication System*, vol. 32, no. 14, pp. e40681–e406832, 2019.

[4] P. Garefalakis, K. Karanasos, P. R. Pietzuch et al., "Scheduling of long running applications in shared production clusters," in *Proceedings of the 13th European Conference on Computer Systems*, Zürich, Switzerland, April 2018.

[5] P. Janus and K. Rzadca, "SLO-aware colocation of data center tasks based on instantaneous processor requirements," in *Proceedings of the 2017 Symposium on Cloud Computing*, Santa Clara, CL, USA, September 2017.

[6] P. K. Sahoo, C. K. Dehury, and B. Veeravalli, "LVRM: on the design of efficient link based virtual resource management algorithm for cloud platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 4, pp. 887–900, 2018.

[7] S. C. Nayak, S. Parida, C. Tripathy, B. Pati, and C. R. Panigrahi, "Multicriteria decision-making techniques for avoiding similar task scheduling conflict in cloud computing," *International Journal of Communication Systems*, vol. 33, no. 13, Article ID e4126, 2019.

[8] L. Lu, J. Yu, Y. Zhu, and M. Li, "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 4, pp. 720–733, 2018.

[9] F. Pascual and K. Rzadca, "Colocating tasks in data centers using a side-effects performance model," *European Journal of Operational Research*, vol. 268, no. 2, pp. 450–462, 2018.

[10] A. Al-Dulaimy, W. Itani, R. Zantout, and A. Zekri, "Type-Aware virtual machine management for energy efficient cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 185–203, 2018.

[11] W. Hou, Z. Ning, L. Guo, Z. Chen, and M. S. Obaidat, "Novel framework of risk-aware virtual network embedding in optical data center networks," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2473–2482, 2018.

[12] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-Aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1385–1400, 2018.

[13] E. Cortez, A. Bonde, A. Muzio et al., "Resource central: understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, Shanghai, China, October 2017.

[14] Y. Cheng, A. Anwar, and X. J. Duan, "Analyzing alibaba's co-located datacenter workloads," in *Proceedings of the 2018 IEEE International Conference on Big Data*, Seattle, WA, USA, Decemeber 2018.

[15] Multi-Criteria Virtual Machine Placement in Cloud Computing Environments: A Literature Review. https://arxiv.org/pdf/1802.05113.pdf.

[16] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1114–1128, 2012.

[17] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *Journal of Lightwave Technology*, vol. 32, no. 3, pp. 450–460, 2014.

[18] J. Hao, B. Zhang, K. Yue et al., "Measuring performance degradation of virtual machines based on the bayesian network with hidden variables," *International Journal of Communication System*, vol. 31, no. 13, Article ID e3732, 2018.

[19] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding LC-VNE algorithms towards integrated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3648–3661, 2016.

[20] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer Societies*, pp. 594–602, Ottawa, Canada, March 1996.

[21] J. Y. Yen, "Finding theKShortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[22] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.