Maximum elastic scheduling of virtual machines in general graph cloud data center networks

Yusuf Qwareeq, Abdalaziz Sawwan & Jie Wu

CYBER-PHYSICAL SYSTEMS
https://doi.org/10.1080/23335777.2023.2301106

Maximum elastic scheduling of virtual machines in general graph cloud data center networks

Yusuf Qwareeq, Abdalaziz Sawwan and Jie Wu

Center for Networked Computing, Science Education and Research Center (SERC), Temple University, Philadelphia, PA, USA

ABSTRACT
In this research, we pioneer a novel method to evaluate the maximum admissible load (MAL) for virtual machines (VMs) in physical machines (PMs) in data centre networks (DCNs), without restricting DCN topologies. This unique approach simplifies the issue into a single-source, multiple-sink maximum flow problem. It also resolves the maximum elastic scheduling problem by determining the optimal load for consistent growth without reassigning tasks. An effective strategy for these challenges is introduced and validated through extensive simulations.

ARTICLE HISTORY
Received 17 February 2023
Accepted 20 November 2023

KEYWORDS
Cloud; data centre networks (DCNs); elasticity; maximum flow; optimisation

1. Introduction

The problem of optimally allocating task resources has always been of utmost importance in cloud-based data centre networks (DCNs). A frequently used model is virtual machine (VM) scheduling in which a certain metric is optimised subject to the capacity of the physical machines (PMs) and links in the DCN [1,2]. Our paper makes use of the *maximum elastic scheduling* scheme [3] to maximise the uniform growth in both computation and communication loads without having to reassign tasks to PMs. This paradigm was originally proposed in [4] for the case of semi-homogeneous trees and further extended in [3] to include heterogeneous trees.

Unlike existing methods that focus exclusively on tree-structured DCNs, our approach broadens the scope to any DCN topology, thus providing a more flexible and efficient architecture for resource allocation. Central to our method is a novel reduction technique that capitalises on the inherent similarities between DCN resource constraints and classical network flow problems. Through the introduction of 'virtual' components – namely, a super source node and virtual switches – we convert the DCN into an analogous flow network. These virtual elements are pivotal: they not only limit the load departing from each processing node according to its capacity but also serve as conduits

CONTACT Yusuf Qwareeq qwareeq@temple.edu Center for Networked Computing, Science Education and Research Center (SERC), Temple University, 332, 1925 N 12th St, Philadelphia, PA 19122, USA

for this load as it transits through the network. This transformation essentially recasts the problem into a single-source, multiple-sink maximum flow problem, enabling us to leverage well-established algorithms in network optimisation to solve for optimal resource allocation.

We model the DCN as an undirected, connected graph $G = (V, E)$ where a node could either be a PM (or processing node) or be a switch. No two PMs are connected directly to each other. The load at a PM is the computation load and it determines the communication load. There are strictly more than two PMs in a DCN. Figure 1 shows a DCN where each PM is represented as a black circle with a number inside it (which is the maximum number of VMs, or computation load, in a PM) while each switch is represented as a grey circle. The numbers associated with the links are the communication bandwidths.

There are mainly two models of communication to connect the VMs in a DCN network. The first model is the *pipe model* [5] in which every PM is connected with all other PMs through separate pipes. This means that the performance guarantees are provided on a per-pipe basis. The second model is the *hose model*, in which every PM is connected to the DCN through a hose, which specifies the total incoming/outgoing bandwidth between the set of all other PMs and that PM. For example, the PM denoted by v_4 in Figure 1 has a computation load of $5B$. This means that the bandwidth v_4 sends to/receives from every other PM in the DCN cannot exceed $5B$. This model provides more flexibility, ease of specification, multiplexing gain, and ease of characterisation along with aggregated performance guarantees per PM [6]. We use the hose model where between each VM and the set of all other VMs, there is a reserved communication bandwidth of B Gbps. There must be no restriction on the destination of the load migrating from one processing node to another.

The *maximum admissible load* (MAL) [3] is defined as the maximum number of VMs whose total communication load can be supported by the underlying structure of the DCN. If a bandwidth of B Gbps is reserved on a link connected

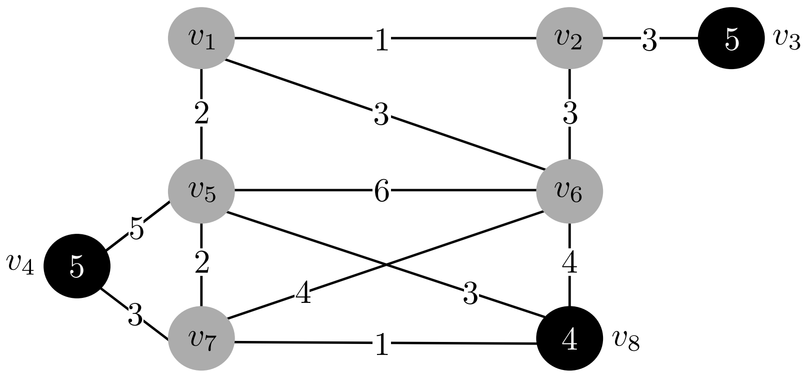


Figure 1. An illustration of a DCN with a general topology. Gray nodes represent switches, while black nodes represent PMs. The number inside each PM represents its computation load.

with a processing node, the bandwidth reserved at that link must equal the contribution of that processing node to the MAL. The schedule with the maximum elasticity (optimal schedule) is the schedule that allows for a uniform increase in the number of VMs across all processing nodes while neither violating the link nor node capacity constraints of the DCN. This elastic scheduling is what forestalls the need for rescheduling tasks when larger loads are to be supported.

Under the hose model, we are concerned with two problems:

- For a DCN of a general topology *G*, given the capacity of its processing nodes and links, what is the MAL of *G*?
- For a load that is admissible, what is the optimal schedule such that the uniform growth rate of the computation load at all processing nodes is maximised under the node and link capacity constraints of the topology?

Another way to look at the two problems is by modelling them as a special utility allocation problem under the hose model in which the processing nodes correspond to households that are connected through a network of cables and switches managed by a telecommunications company. Each one of the households has a certain limit on the number of telephone communications that could be done simultaneously, which corresponds to its occupancy limit. Each cable has a certain communication bandwidth capacity. There is no constraint on the underlying structure of the topology of the network and there must be no restriction on which household is connected to which. Given those settings, what is the bandwidth of all possible simultaneous pairwise telephone conversations that the network can support? Furthermore, what is the communication bandwidth assignment at each household and cable such that all capacity constraints are satisfied?

In [3], the MAL and maximum elasticity are found for tree-structured DCNs using a distributed, linear-time algorithm that goes over the tree level-by-level through two steps. First, a bottom-up aggregation step determines the MAL of the tree. Second, a top-down partitioning step determines the schedule with the maximum elasticity. However, the supported topologies of the DCNs are exclusive to tree structures. In contrast, our work is generalised over any DCN topology. Additionally, the processing nodes are free to exist in any part of the graph with as many links connecting them to the rest of the DCN as necessary. This provides flexibility in the architecture of the DCN such that processing nodes can be deployed to any place within the network.

Our results in this research study are summarised as follows:

- We show that the more generalised version of the problem (no constraints on the topology) can be reduced to a single-source, multiple-sink maximum flow problem.
- We prove the validity of this reduction and its consequences on answering the two broader problems at hand.
- We present a novel algorithm that provides the general framework which can utilise different existing algorithms that solve the single-source, multiple-sink maximum flow problem.
- We evaluate the efficiency of our solution through an extensive simulation.

The remainder of the paper has the following organisation. Section 2 presents the preliminaries and the formulation of the problem. Section 3 introduces our optimal algorithm for finding the MAL and the schedule with the maximum elasticity before proving its optimality. Section 4 shows our experiments and simulation results. Section 5 discusses related work. Finally, the paper is concluded in Section 6.

2. Problem formulation

2.1. Preliminaries

We consider the DCN under the hose model where the quality of service requirements are specified per processing node (unlike the pipe model where the requirements are specified per pair of processing nodes) [6]. The processing nodes are connected to the DCN with assigned ingress and egress bandwidths where each bandwidth defines how much of the load could be sent to/from each processing node from/to every other processing node. The two are assumed to be equal.

Figure 2 illustrates the difference between the pipe model and the hose model. In order to connect node C under the pipe model, there must exist separate pipes connecting it to the two other nodes, A and B. Hence, the total bandwidth reserved for node C will be $2B + 3B = 5B$. In other words, it is the direct sum of the bandwidths of the two other nodes. On the contrary, under the hose model, there needs to be only one pipe that supports the aggregated bandwidth of the two other nodes, which is $\max\{2B, 3B\} = 3B$.

Furthermore, the PMs must not be directly connected. To establish a connection between two VMs in different processing nodes at any given time, there must be a path of capacity B Gbps in the DCN reserved for this connection at that time. This means that generally speaking, the sum of the link capacities leaving a PM must be large enough to withstand the total communication load of the VMs leaving that PM when simultaneous connections of multiple VMs in the same processing node to the DCN are established.

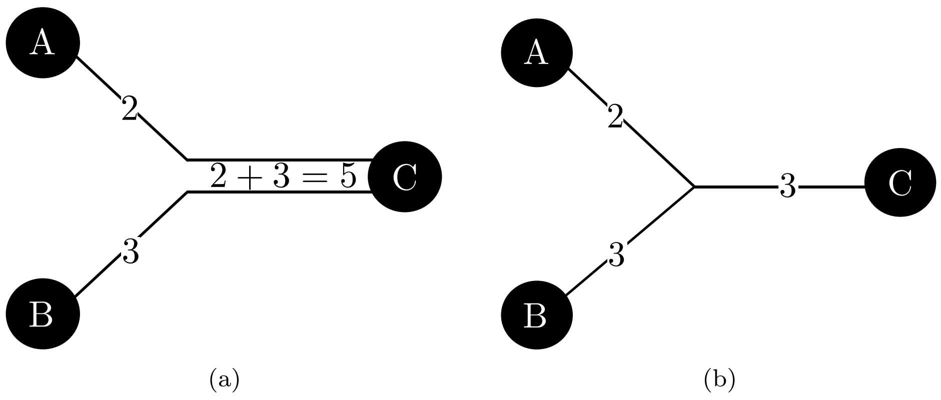


Figure 2. The difference between (a) the pipe model and (b) the hose model.

2.2. The problem

The paper aims to identify the MAL and schedule with the maximum elasticity for a cloud DCN. A DCN is modelled as an undirected, connected graph $G = (V, E)$ where $V' \subset V$ is the set of DCN PMs (processing nodes) with computation loads $\{L_{v'} | v' \in V'\}$ (maximum number of VMs in a PM), $V \backslash V'$ is the set of DCN switches interconnecting the network elements, and E is the set of undirected edges connecting the switches with other switches and processing nodes, which have capacities $\{C_e | e \in E\}$.

3. Optimal solution

3.1. Algorithm overview

The breadth of our algorithm lies in reducing the problems of finding the MAL and the schedule with the maximum elasticity to a single-source, multiple-sink maximum flow problem. First, we augment the graph G by adding a node (termed a *virtual switch*) for each PM and attaching it to the PM. The capacity of the edge that connects these two nodes (coined a *virtual hose*) is equal to the processing node capacity $L_{v'}$ of that PM. We then swap the virtual switch with its corresponding PM. After doing this for every PM, we coalesce the set of all PMs V' into a single super node which we call the source node $\mathcal{S}$. After that, we solve multiple maximum flow problems with $\mathcal{S}$ as the source node while choosing the sink as a different node from $V \backslash V'$ at every instance, and we store the result. This result is the admissible load that G can support given a certain source-sink pairing. The maximum of those results is the MAL. Algorithm 1 shows the pseudocode of our strategy that evaluates the MAL of a DCN represented by G. Regarding the schedule with the maximum elasticity, it can be directly obtained using the proportion of the flow values through the virtual hoses given an MAL assignment. Allocating the total load on the processing nodes with that proportion results in the maximum elastic schedule that allows for the largest possible uniform growth in the computation loads across the processing nodes.

Algorithm 1 Evaluating the MAL

```
Input: $G = (V, E)$, $V' \in V$, $\{L_{v'} | v' \in V'\}$, $A$.
Output: The MAL value.
Initialisation: $A = \{\}$ // Set of admissible loads.
1: for each $v'$ in $V'$ do
2:     Add to $G$ a node $i_{v'}$ and a link $(i_{v'}, v')$ of capacity $L_{v'}$.
3:     Swap node $i_{v'}$ with node $v'$.
4: Coalesce $V'$ into a single super source node $\mathcal{S}$.
5: for each $v$ in $V \backslash V'$ do
6:     $A \leftarrow \mathrm{MAX}-\mathrm{FLOW}_G(\mathcal{S}, v)$ // Append to set $A$.
7: return $\max(A)$.
```

The algorithm can now be demonstrated through an example. Figure 1 shows a DCN with 3 processing nodes ($V' = \{v_3, v_4, v_8\}$) and 5 switches ($V \backslash V' = \{v_1, v_2, v_5, v_6, v_7\}$). Our solution revolves around the idea of reducing the problem of finding the MAL to a maximum flow problem. To do this, for each v' in V', we add a virtual switch $i_{v'}$ with a virtual hose that connects it to v'. The capacity of this virtual hose is equal to the computation load of v' which is $L_{v'}$. We then swap every virtual switch $i_{v'}$ with its corresponding v'. After that, nodes in V' are coalesced into a single node which we treat as a super source node $\mathcal{S}$.

The maximum flow is then applied in a one-to-all manner between $\mathcal{S}$ and each v in $V \backslash V'$. From there on, the set of solutions A that we get is the set of admissible loads given load assignments rooted at each one of the switches. The admissible load with the maximum value will be the MAL. This process is described in Algorithm 1. The pseudocode describing the algorithm starts with determining the input, which is the DCN topology encoded in a graph so that each node in the graph is specified to be either a switch or a processing node. Furthermore, the values of the capacities of the links are determined alongside the values of the processing node capacities. The output of the algorithm will be the MAL.

Now, starting with the initialisation, the set of admissible loads A is reserved. Lines 1–3 of the algorithm replace each processing node v' with a virtual switch $i_{v'}$, a virtual hose $(i_{v'}, v')$, and v' itself. The capacity of $(i_{v'}, v')$ is set to $L_{v'}$. This makes each processing node v' connected to the graph through exactly one link $(i_{v'}, v')$. Figure 3 shows the augmented G after applying these lines to the DCN in Figure 1. Line 4 of the algorithm coalesces the processing nodes into one node

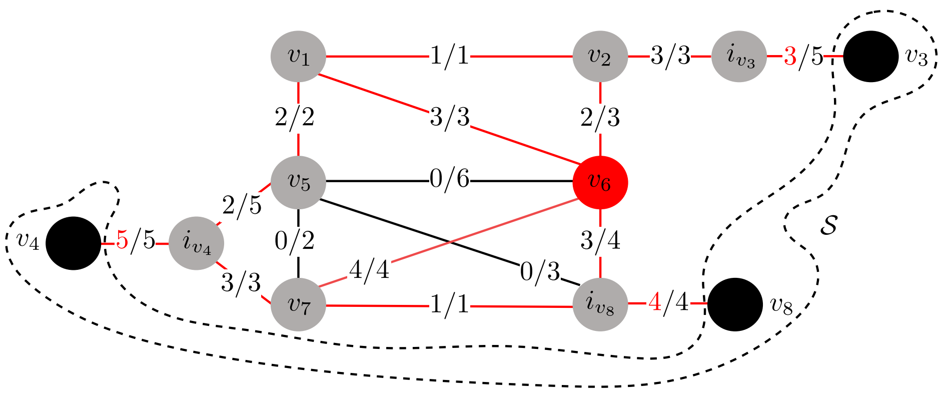


Figure 3. An example of evaluating the MAL for one source-sink pairing of the DCN shown in Figure 1 (taking super node $\mathcal{S}$ as the source and node v_6, which is marked in red, as the sink).

Table 1. The set of admissible loads A found after applying $\text{MAX} - \text{FLOW}_G(\mathcal{S}, v)$ to every node $v \in V \backslash V'$ in the DCN in Figure 3.

Node Taken as Sink	v_1	v_2	v_5	v_6	v_7
Admissible Load (B)	6	7	12	12	10

called the super source node, denoted as $\mathcal{S}$. Lines 5–6 apply the one-to-all maximum flow algorithm in the most primitive way possible by applying a one-to-one maximum flow algorithm $|V \backslash V'|$ times considering S as the source and a different node from $|V \backslash V'|$ as the sink in each maximum flow instance. The last line returns the largest value of the maximum flows generated. This value is the target MAL.

Table 1 shows the list of admissible loads A after applying Algorithm 1 to the DCN in Figure 1. The MAL for the example is $12B$ and the schedule with the maximum elasticity is the one that follows the proportion of the flow values through the virtual hoses (the numbers are colored in red) in the figure ($3 : 5 : 4$ for nodes v_3, v_4, and v_8, respectively). This proportion is different from the PM's available capacities. If we were to try any different schedule, the maximum uniform growth possible in the computation loads would decrease. This means that, given the same topology with the same (or linearly scaled) values assigned to the link and processing node capacities, the optimal assignment in order for the load on the PMs to have the maximum growth rate must follow the proportion $3 : 5 : 4$.

We may now investigate the schedule with the maximum elasticity. For example, given the same topology with all the values of link and processing node capacities multiplied by 20, and given that the total load assignment allocated for all the processing nodes is predetermined to be $84B$, the most elastic assignment for this load on the processing nodes must follow the proportion $3 : 5 : 4$. Hence, the optimal load assignment of the predetermined load will be $21B$, $35B$, and $28B$ for nodes v_3, v_4, and v_8, respectively. Having this assignment yields the optimal elasticity value of $(60 + 100 + 80)/(21 + 35 + 28) = 285.7\%$. However, if we, for example, consider the load proportion to be the same as the one of the processing node capacities (which is $L_{v_3} : L_{v_4} : L_{v_8} = 5 : 5 : 4$) such that the load is assigned to the processing nodes with values of $30B$, $30B$, and $24B$ for nodes v_3, v_4, and v_8, respectively, the elasticity value will now be $(60 + 60 + 48)/(30 + 30 + 24) = 200.0\%$, which is less than the optimal elasticity value.

3.2. Algorithm analysis

In this subsection, we prove the optimality of Algorithm 1, show that it provides the schedule with the maximum elasticity, and find its time complexity.

Theorem 1. *The largest maximum flow evaluated for all possible pairings of the super source node $\mathcal{S}$ and every node $v \in V \backslash V'$ is the MAL of the graph G.*

Proof. Given a graph $G = (V, E)$, we first demonstrate that admissible loads are analogous to classical feasible flow assignments. We begin by examining the applicability of the definition of feasible flows to admissible loads after adding the virtual switches and the virtual hoses with capacities equal to their corresponding processing node capacities. A virtual hose of capacity $L_{v'}$ connected between the coalesced super source node $\mathcal{S}$ and the virtual switch $i_{v'}$ serves as a cap equal to the capacity of the processing node v' for the amount of load going out from it, which is determined by how many VMs the PM can support concurrently.

Feasible flows are simply defined as abstract quantities assigned to each edge of s–t networks (where s is the source node and t is the sink node) while satisfying two properties. The first one is the capacity condition, which means that for each $e \in E$, we have $0 \leq f(e) \leq C_e$, where an s–t flow is a function f that maps each edge e to a non-negative integer, $f : E \rightarrow \mathbb{N}^+$; the value $f(e)$ intuitively represents the amount of flow carried by edge e. This first condition identically applies to loads going through links in DCNs as loads may not exceed the capacity of the links carrying them.

The second property is the conservation condition, which states that for each node v other than s and t, we have $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$. Now, for an MAL assignment, since the bandwidth reserved at the links connected with each processing node equals the contribution of that processing node to the MAL, and there is no constraint on the choice of the destination processing node for a load going out from another processing node, and there is strictly more than two processing nodes in the DCN, there will be at least one switch through which all of the communication load between the processing nodes passes. This switch plays the role of a sink node t while coalesced processing nodes (or the super source node $\mathcal{S}$) are modelled as the source node s.

Under the MAL assignment, the load can be thought of as a flow going out from the processing nodes, that are modelled as s, into the switch, which is modelled as t. Since the entirety of this MAL is leaving s, draining in t, and conserved when it goes through any other switch, the second property of feasible flows applies to MAL assignments; both are considered conserved quantities constrained by the capacities of the links of the graph that they pass through. Hence, evaluating the MAL is analogous to finding the maximum flow when $\mathcal{S}$ is considered to be s, and the correct switch, which has all of the load going through it, is considered to be t.

Lastly, we need to specify that correct switch under the MAL assignment. Since we have already shown that such a switch always exists, searching for it is done by exhausting all the switches in the DCN. Once that switch is found, choosing it as t would guarantee finding the MAL value. ▪

Note that it is possible for a DCN to be able to support multiple different load assignments under the same MAL, as more than one switch may yield the MAL value when chosen as the sink node t. In addition, even for the same switch chosen as a sink, there might be different valid MAL assignments. The assignment our algorithm produces depends on the method chosen to evaluate the maximum flow.

Theorem 2. *Assigning a predetermined load to the processing nodes with the same proportion of the loads under the MAL assignment results in the schedule with the maximum elasticity.*

Proof. Following the proportion of the loads under the MAL assignment when setting the loads across the processing nodes maximises the elasticity because this is the only proportion of loads which guarantees that all of them will reach their maximum potential together, given a uniform linear growth in the loads of the processing nodes. Any deviation from said proportion would cause at least one of the processing nodes to reach its bottleneck before others reach theirs. ▪

The time complexity of Algorithm 1 is $O(|V \backslash V'| \times T(|V|))$ where $T(|V|)$ is the time complexity of one instance of the one-to-one maximum flow algorithm. After coalescing the PMs, since we perform the one-to-one maximum flow between the coalesced nodes (super source node $\mathcal{S}$) and every switch in the graph, we end up invoking the one-to-one maximum flow algorithm $|V \backslash V'|$ times (which is the number of switches in G), yielding the aforementioned time complexity.

Although we opted to choose the most basic way to perform the one-to-all maximum flow algorithm, there have been extensive studies on more efficient ways to do it. For example, Lacki *et al.* [7] have come up with a more efficient algorithm that would perform the complete one-to-all maximum flow in $O(|V \backslash V'| \times \log^3 |V \backslash V'|)$ time in case the graph is planar.

On the other hand, if we stick to the basic strategy and use the principal Ford – Fulkerson algorithm [8] $|V \backslash V'|$ times, we get a total time complexity of $O(|V \backslash V'| \times |E| \times |\mathrm{MAL}|)$. We can use Dinic's algorithm [9] (which is an improved version of the Edmonds – Karp algorithm [10]) if we desire a time complexity independent from the value of the MAL. That would yield a total time complexity of $O(|V \backslash V'|^3 \times |E|)$.

The most efficient $T(|V|)$ time that has been developed for a one-to-one maximum flow algorithm applied for the case of a general graph is the one introduced by Chen *et al.* [11]. Their new one-to-one maximum flow strategy asymptotically outperforms any existing algorithm with a time complexity of $O(|E|^{1+o(1)})$. If we choose their algorithm for our $\mathrm{MAX}-\mathrm{FLOW}_G(\mathcal{S}, v)$ subroutine, the time complexity of our strategy will reduce to $O(|V \backslash V'| \times |E|^{1+o(1)})$. Finally, Abboud *et al.* [12] have come up with the best all-pair maximum flow algorithm in terms of the time complexity bound. Their algorithm can be applied to evaluate the MAL in our case with a time complexity of $O(|E|^{3/2+o(1)})$.

4. Simulation

We use NetworkX [13] to generate graphs randomly by first creating a k-node spanning tree from a Prüfer sequence [14]. A Prüfer sequence from the k possible sequences is chosen at random and then inputted into a Prüfer decoding algorithm that produces a unique tree T. After that, the switches are added in a one-by-one manner to create an undirected, connected graph G with $|V \backslash V'|$ nodes. When adding a switch, a normal random variable $X \sim \mathcal{N}(\mu_e, \sigma_e^2)$ is sampled to decide the number of edges connecting that switch to G. After adding the switches and their edges, a fraction of the total number of nodes ($\in |V|$) is chosen as processing nodes V' and are then added in a one-by-one manner as well. The number of edges connecting them to G is picked in a similar fashion to switches but we make sure these processing nodes are not directly linked to each other. From there on, a super source node $\mathcal{S}$ is connected to each processing node with an edge that has a capacity sampled from the uniform random variable $Y \sim \mathcal{N}(\mu_{L_{V'}}, \sigma^2_{L_{V'}})$. As for the other edges, their capacities are sampled from the uniform random variable $Z \sim \mathcal{N}(\mu_{C_E}, \sigma^2_{C_E})$.

4.1. Algorithm comparison

We consider various settings to compare the performance of four algorithms: our optimal algorithm (Algorithm 1), Proportion with Physical Link Capacities (PPLC) [3] where the VMs are assigned into the processing nodes with proportion to the bandwidth (link capacities), Equally Distributed Placement (EDP) where the VMs are evenly assigned into the processing nodes, and Maximum Spanning Tree (MST) where we build a maximum spanning tree on G using Kruskal's algorithm [15]. In terms of time complexities, the EDP algorithm, which uniformly apportions loads across the PMs, has a time complexity of $O(V')$. The PPLC algorithm, on the other hand, ascertains the aggregate bandwidth of links affiliated with each PM, and iteratively examines both PMs and their associated links, leading to an average time complexity of $O(\mu_e \times V')$. Lastly, the MST algorithm first constructs a maximum spanning tree on G utilising Kruskal's algorithm, which incurs a time complexity of $O(E \log V)$, followed by a solution that is constrained to tree topologies, which takes a complexity of $O(V)$ [3]. Cumulatively, MST exhibits a time complexity of $O(E \log V + V)$.

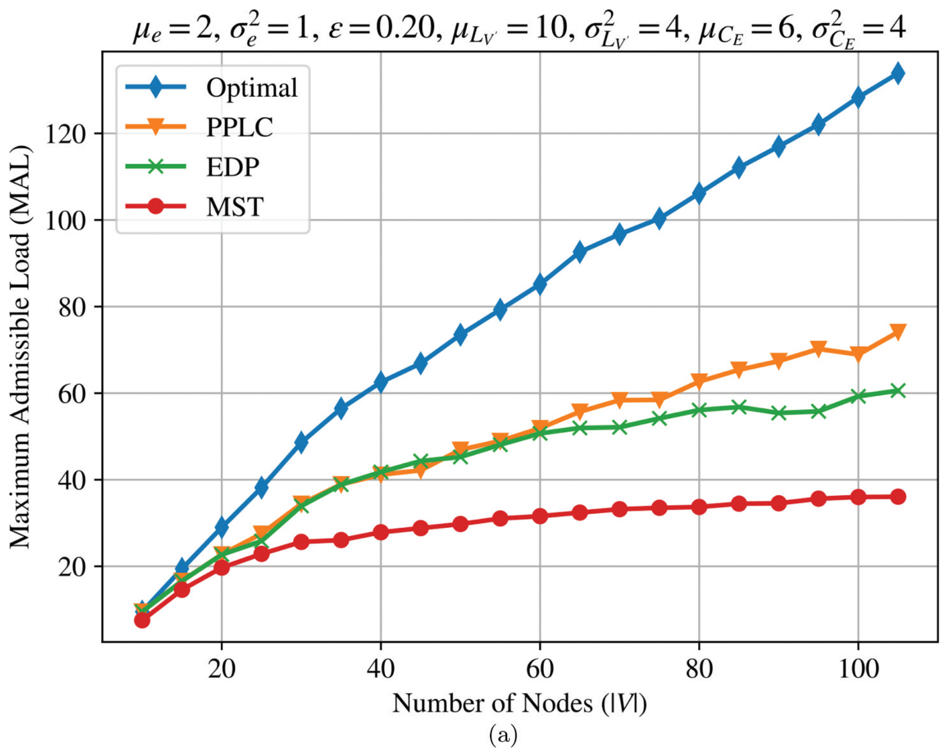
4.2. Experimental results

In subfigures (a) and (b) of Figure 4, the number of nodes $|V|$ was varied for a sparse and dense network, respectively. Notice how increasing the number of nodes increased the MAL up to a certain point where the MAL tapered off. This happened because $\in$ and $\mu^2_{L_{V'}}$ are fixed, and thus, a bottleneck in how much flow G could support is formed. The MAL of the optimal algorithm would take the longest time to saturate and is consistently higher than the MAL of the other algorithms. We can also see how the bottleneck is less of an issue in (b) when compared to (a) due to the difference in D.

On the other hand, Figure 5's subfigures (a) and (b) show how changing the number of processing nodes $|V'|$ affects the MAL for a sparse and dense network, respectively. Note how increasing the number of processing nodes always increases the MAL the DCN is able to withstand. However, the optimal algorithm fully makes use of the limitations of G which is why it outperforms the other algorithms for the same mean of processing node capacities. Additionally, due to the increase in D when comparing (b) to (a), the bottleneck that exists in the link capacities μ_{C_E} becomes a non-issue due to the huge number of edges in G.

Further, subfigures (a) and (b) of Figure 6 show the effect of varying the mean of the processing node capacities $\mu_{L_{V'}}$. The subfigures are split into two regions based on whether the link capacities or the processing node capacities are the dominant factor. To the left of the knee of the two subfigures, the processing node capacities are what is limiting the MAL. The MAL continues to increase as the processing node capacities are increased until the edge capacities become unable to support the flow leaving the processing nodes causing the MAL to hit saturation. This limit is higher when comparing (b) to (a) due to the increase in the number of links.

Finally, Figure 7's subfigures (a) and (b) show how varying the mean of the link capacities μ_{C_E} influences the MAL for a sparse and dense network,



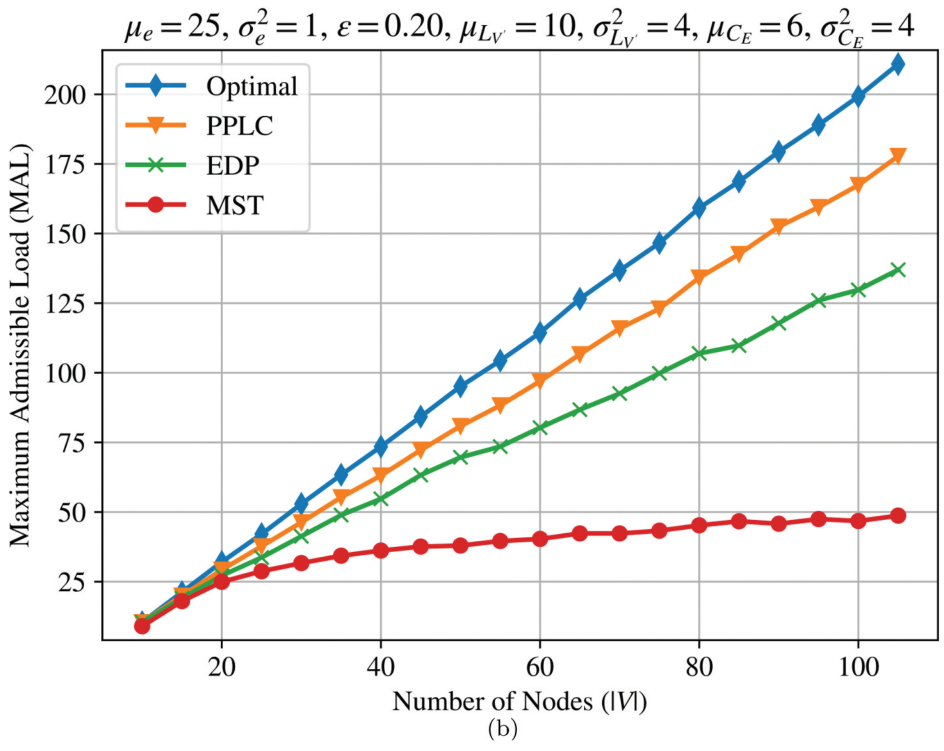
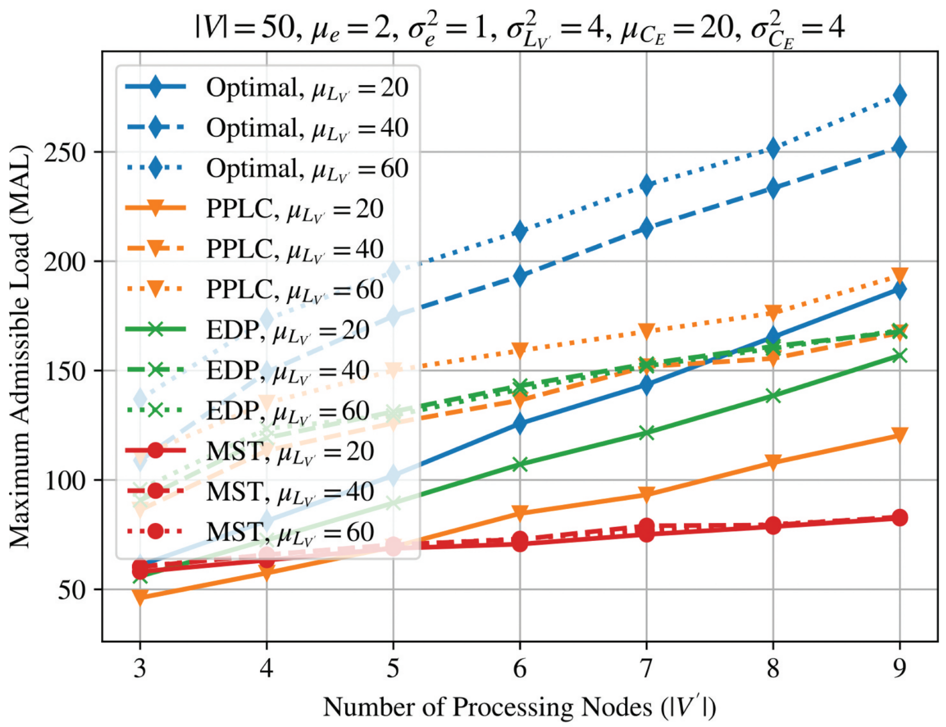
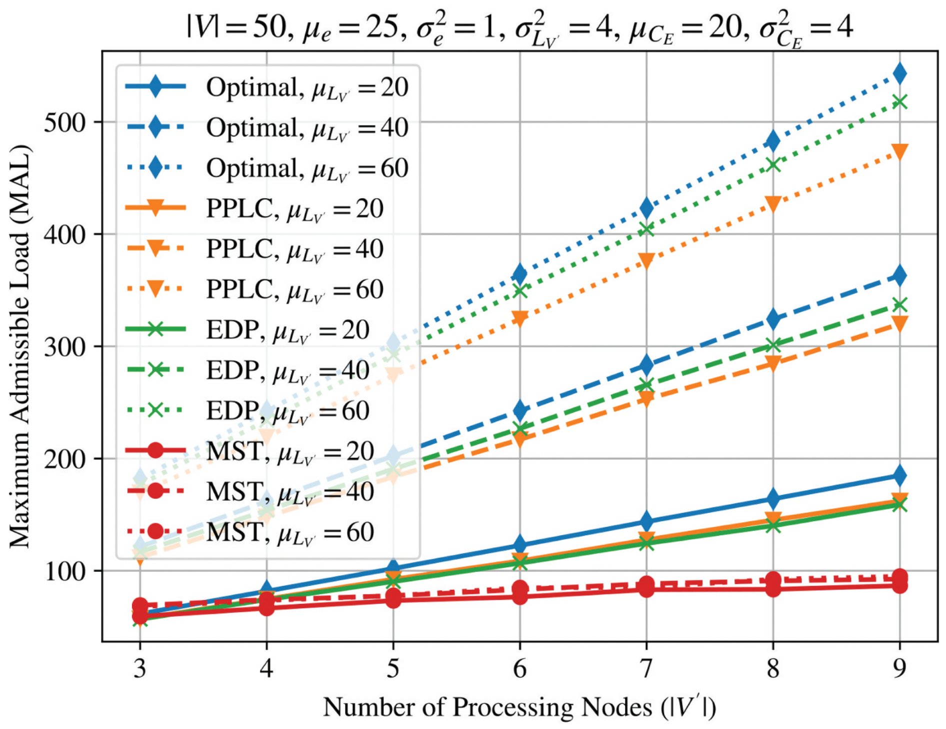


Figure 4. How the MAL is affected due to varying the number of nodes $|V|$ in (a) a sparse network and (b) a dense network during the generation process of the random graphs.

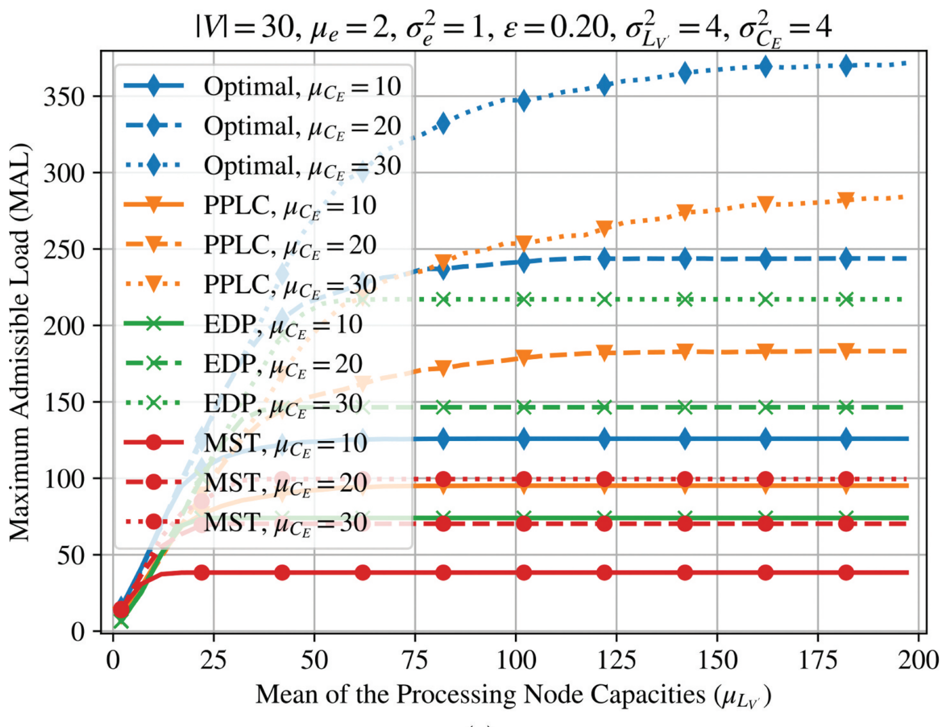


(a)

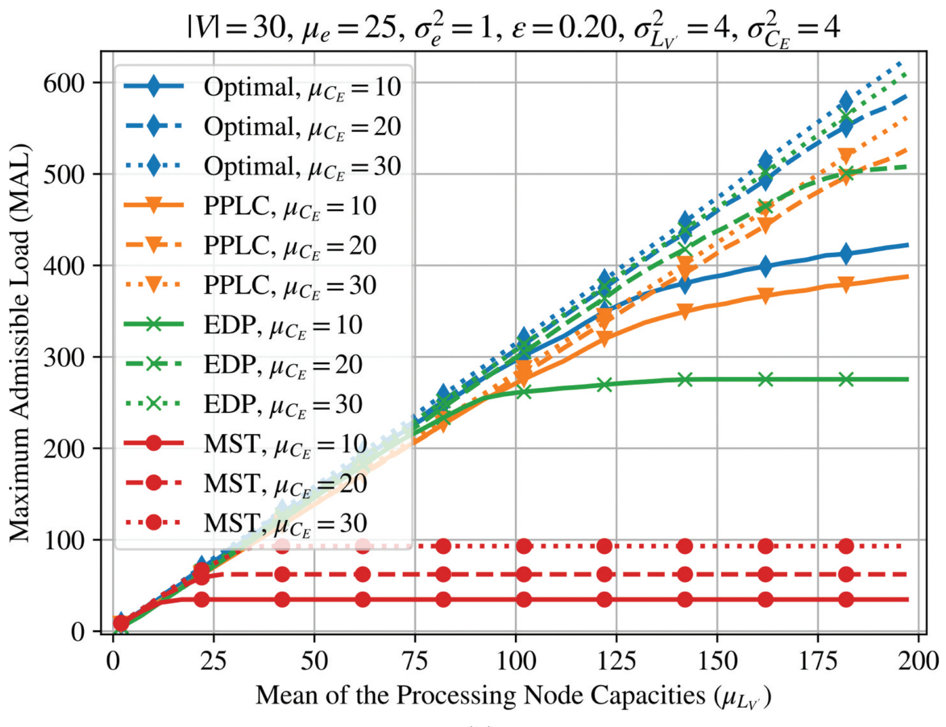


(b)

Figure 5. How the MAL is affected due to varying the number of processing nodes $|V'|$ in (a) a sparse network and (b) a dense network during the generation process of the random graphs.

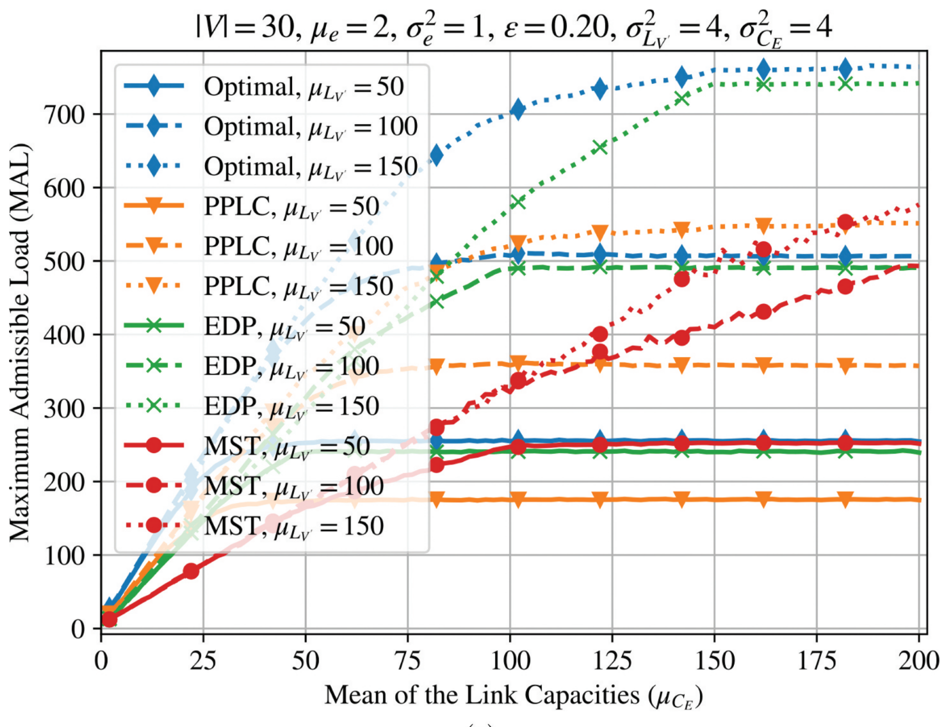


(a)

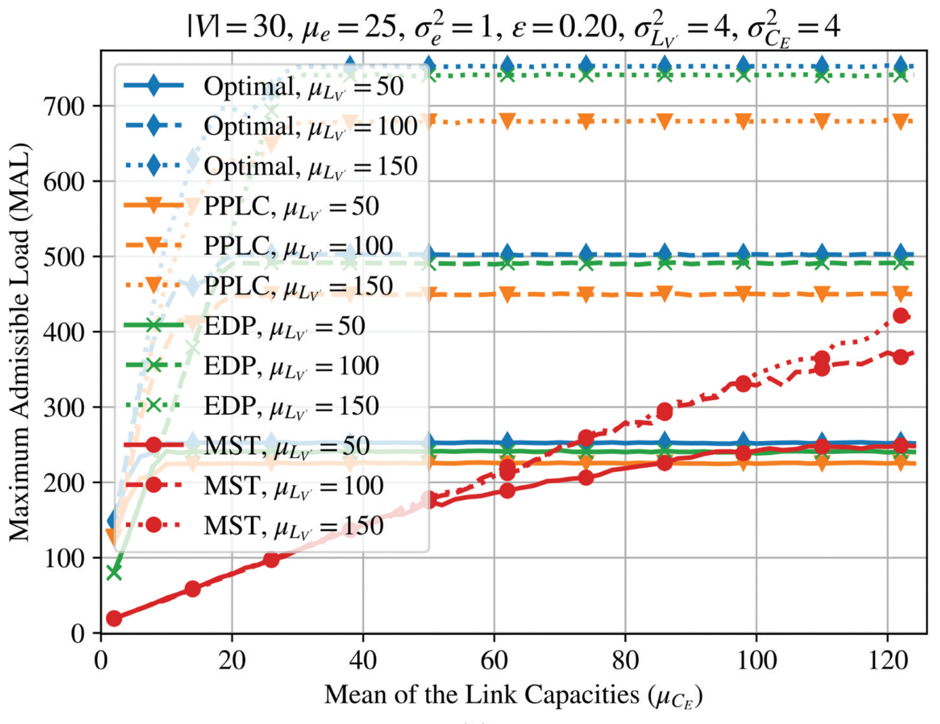


(b)

Figure 6. How the MAL is affected due to varying the mean of the processing node capacities $\mu_{L_{V'}}$ in (a) a sparse network and (b) a dense network during the generation process of the random graphs.



(a)



(b)

Figure 7. How the MAL is affected due to varying the mean of the link capacities μ_{C_E} in (a) a sparse network and (b) a dense network during the generation process of the random graphs.

respectively. We have a similar story to the other two subfigures in Figure 6, except this time, the two regions are swapped. The region to the left of the knee of the curves is dominated by the link capacities while the region to the right of the knee of the curves is dominated by the processing node capacities. Having a dense network this time allows the network to reach saturation earlier.

5. Related work

VM scheduling is a widely used technology in order to guarantee application isolation and to simultaneously allow for utilisation of the PMs. Much has been accomplished in VM placement in cloud-based DCNs [1,2,16] with constraints including power [17], reliability [18], and traffic minimisation [19]. Other practical factors in VM scheduling have also been discussed [20,21]. Predictability is another important goal in designing efficient DCNs [22,23]. Empirical estimates of bandwidth are often used in allocating computing and network resources [24,25].

Elasticity, or the ability to adapt to changes in workload, is a central aspect of cloud computing. It is crucial when estimation of resources cannot be easily obtained [3,26–28]. In this context, the authors in [4] pioneered the concept of elasticity specifically for VMs in datacenters. They introduced a hierarchical VM placement algorithm that considers both machine and bandwidth resources, proving its optimality in semi-homogeneous and K-ary datacenter topologies. Their evaluation results demonstrated the efficiency of their approach in multi-tenant datacenters. Cloud computing elasticity is also defined as the degree to which a system autonomically provisions and de-provisions resources to adapt to workload changes [29]. Shawky and Ahmed have provided benchmarks for measuring this attribute [26].

The concept of maximum elasticity scheduling was explored by the authors in [3]. They focused on optimising task resource allocation in tree-based cloud data centre networks. They introduced the hose model for communication and discussed the limitations of static load distribution. To circumvent these issues, they proposed a maximum elasticity scheduling method, revealing that it has the highest growth potential subject to node and link capacities. Their findings generalise the one-to-all maximum flow problem, positioning their work as a special case of this broader issue.

6. Conclusion

Task resource assignment has generated a lot of research activity in the applications of cloud-based DCNs. In this study, we introduced an optimal solution that computed the MAL given a number of VMs inside PMs. This is the first work that does not place any constraints on the topologies of the DCNs. We did this by proving that this problem could be reduced to a single-source, multiple-sink maximum flow problem. Additionally, we found the optimal VM assignment such that the maximum possible uniform growth rate could be supported without having to reassign tasks to PMs, known as the maximum elastic schedule. Our approach's efficiency and effectiveness were validated via extensive simulations, comparing it to three different algorithms.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was supported in part by NSF grants CPS 2128378, CNS 2107014, CNS 2150152, CNS 1824440, CNS 1828363, and CNS 1757533.

Notes on contributors

Yusuf Qwareeq, is a first-year PhD candidate in the Computer & Information Sciences department at Temple University. He holds a Bachelor of Science in Electrical Engineering from the University of Jordan, which he completed in 2021. Qwareeq's current research interests include networks, distributed computing, and parallel processing.

Abdalaziz Sawwan, is a second-year Ph.D. student in Computer and Information Sciences at Temple University. Sawwan received his bachelor's degree in Electrical Engineering from the University of Jordan in 2020. His current research interests include multi-armed bandits, communication networks, mobile charging, and wireless networks.

Jie Wu, is a Laura H. Carnell Professor and the Director of the Center for Networked Computing at Temple University. He also serves as the Director of International Affairs at the College of Science and Technology. Dr. Wu was previously the Chair of the Department of Computer and Information Sciences and Associate Vice Provost for International Affairs at Temple University. Prior to joining Temple University, he was a distinguished professor at Florida Atlantic University and a program director at the National Science Foundation. Dr. Wu's research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He has published extensively in scholarly journals, conference proceedings, and books, and serves on several editorial boards. Dr. Wu has also chaired and co-chaired several IEEE and ACM conferences and is a Fellow of both the IEEE and the AAAS.

Data availability statement

Data sharing is not applicable to this article as no new data were created or analysed in this study.

References

[1] Á Mann Z. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput Surveys. 2015;48(1):1–34. doi:10.1145/2797211

[2] Saif MAN, Niranjan S, Al-Ariki HDE. Efficient autonomic and elastic resource management techniques in cloud environment: taxonomy and analysis. Wireless Networks. 2021;27(4):2829–2866. doi:10.1007/s11276-021-02614-1

[3] Wu J, Lu S, Zheng H. On maximum elastic scheduling of virtual machines for cloud-based data center networks. 2018 IEEE International Conference on Communications (ICC); Kansas City, MO, USA. IEEE; 2018. p. 1–6.

[4] Li K, Wu J, Blaisse A. Elasticity-aware virtual machine placement for cloud datacenters. 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet); San Francisco, CA, USA. IEEE; 2013. p. 99–107.

[5] Duffield NG, Goyal P, Greenberg A, et al. A flexible model for resource management in virtual private networks. Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication; New York, NY, United States. 1999. p. 95–108.

[6] Kumar A, Rastogi R, Silberschatz A, et al. Algorithms for provisioning virtual private networks in the hose model. IEEE/ACM Trans Networking. 2002;10(4):565–578. doi:10.1109/TNET.2002.802141

[7] Lacki J, Nussbaum Y, Sankowski P, et al. Single source–all sinks max flows in planar digraphs. 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science; New Brunswick, NJ, USA. IEEE; 2012. p. 599–608.

[8] Ford LR, Fulkerson DR. Flows in networks. In: Flows in networks. Princeton, NJ, USA: Princeton university press; 2015 .

[9] Dinic E. Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math Doll. 1970;11(5):1277–1280. English translation by RF. Rinehart, 1970.

[10] Edmonds J, Karp RM. Theoretical improvements in algorithmic efficiency for network flow problems. J ACM. 1972;19(2):248–264. doi:10.1145/321694.321699

[11] Chen L, Kyng R, Liu YP, et al. Maximum flow and minimum-cost flow in almost-linear time. 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS); Denver, CO, USA. IEEE; 2022. p. 612–623.

[12] Abboud A, Krauthgamer R, Trabelsi O. New algorithms and lower bounds for all-pairs max-flow in undirected graphs. Proceedings of the Fourteenth Annual ACMSIAM Symposium on Discrete Algorithms; Salt Lake City, UT, USA. SIAM; 2020. p. 48–61.

[13] Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T Millman J, editors. Proceedings of the 7th Python in Science Conference, Pasadena, CA USA; 2008, p. 11–15.

[14] Wu BY, Chao K-M. Spanning trees and optimization problems. Boca Raton, FL, USA: Chapman and Hall/CRC; 2004.

[15] Pemmaraju S, Skiena S. Computational discrete mathematics: combinatorics and graph theory with mathematica®. Cambridge, UK: Cambridge university press; 2003.

[16] Sridharan R, Domnic S. Network policy aware placement of tasks for elastic applications in iaas-cloud environment. Cluster Comput. 2021;24(2):1381–1396. doi: 10.1007/s10586-020-03194-z

[17] Kusic D, Kephart JO, Hanson JE, et al. Power and performance management of virtualized computing environments via lookahead control. Cluster Comput. 2009;12(1):1–15. doi:10.1007/s10586-008-0070-y

[18] Yang S, Wieder P, Yahyapour R, et al. Reliable virtual machine placement and routing in clouds. IEEE Trans Parallel Distrib Syst. 2017;28(10):2965–2978. doi:10.1109/TPDS.2017.2693273

[19] Meng X, Pappas V, Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement. 2010 Proceedings IEEE INFOCOM; San Diego, CA, USA. IEEE; 2010. p. 1–9.

[20] Xu F, Liu F, Jin H, et al. Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc IEEE. 2013;102(1):11–31. doi:10.1109/JPROC.2013.2287711

[21] López-Pires F, Barán B. Cloud computing resource allocation taxonomies. Int J Cloud Comput. 2017;6(3):238–264. doi:10.1504/IJCC.2017.086712

[22] Ballani H, Costa P, Karagiannis T, et al. Towards predictable datacenter networks. Proceedings of the ACM SIGCOMM 2011 Conference; New York, NY, USA. 2011. p. 242–253.

[23] Silva Filho MC, Monteiro CC, Inácio PR, et al. Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. J Parallel Distrib Comput. 2018;111:222–250. doi:10.1016/j.jpdc.2017.08.010

[24] Wang R, Wickboldt JA, Esteves RP, et al. Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters. IEEE Trans Network Serv Manage. 2016;13(2):267–280. doi:10.1109/TNSM.2016.2530309

[25] Laghrissi A, Taleb T. A survey on the placement of virtual resources and virtual network functions. IEEE Commun Surv Tutorials. 2018;21(2):1409–1434. doi:10.1109/COMST.2018.2884835

[26] Shawky DM, Ali AF. Defining a measure of cloud computing elasticity. 2012 1st International conference on systems and computer science (ICSCS); Lille, France. IEEE; 2012. p. 1–5.

[27] Lu S, Fang Z, Wu J, et al. Elastic scaling of virtual clusters in cloud data center networks. 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC); San Diego, CA, USA. IEEE; 2017. p. 1–8.

[28] Li K, Wang L. Elastic scheduling of virtual machines in cloudlet networks. 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC); Austin, TX, USA. IEEE; 2021. p. 1–7.

[29] Herbst NR, Kounev S, Reussner R. Elasticity in cloud computing: what it is, and what it is not. 10th international conference on autonomic computing (ICAC 13); San Jose, CA. 2013. p. 23–27.