Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 11, No. 3, August 2020), Pages: 1-21
www.jacr.iausari.ac.ir

Communication-Aware Traffic Stream Optimization for Virtual Machine Placement in Cloud Datacenters with VL2 Topology

Sara Farzai[1], Mirsaeid Hosseini Shirvani[1] *, Mohsen Rabbani[2]
1) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
2) Department of Mathematics, Sari Branch, Islamic Azad University, Sari, Iran
sara.farzai@gmail.com; mirsaeid_hosseini@iausari.ac.ir; mrabbani@iausari.ac.ir

Received: 2019/09/05; Accepted: 2020/09/20

Abstract

By pervasiveness of cloud computing, a colossal amount of applications from gigantic organizations increasingly tend to rely on cloud services. These demands caused a great number of applications in form of couple of virtual machines (VMs) requests to be executed on data centers' servers. Some of applications are as big as not possible to be processed upon a single VM. Also, there exists several distributed applications such as MapReduce projects which exploit much number of VMs dispersed over physical machines (PMs) attached with high speed networks. These types of VMs involve mutual traffic transferring which is completely processed as an atomic application. High volume of traffic transfer among VMs may saturate network links and leads performance bottleneck for both data center and applications which seriously threat users' service level agreement (SLA). Furthermore, communication energy consumption increases when network devices are heavily in use. This paper addresses the virtual machine placement (VMP) problem by considering inter-VM communications on VL2 topology. This is an optimization problem with the aim of network traffic transferring minimization. Dependent VMs are tried to be co-hosted or to be placed in close neighborhoods to minimize the amount of total traffic streaming over the network. A combined meta-heuristic approach based and ACO and GA algorithms is employed to solve the problem. The results of simulations imply the superiority of our proposed approach in comparison with other state-of-the-art approaches in terms of reducing total traffic flow, saving energy, and declining resource dissipation in servers.

Keywords: Cloud Computing, Network Traffic Management, Virtual Machine Placement, VL2, Meta-Heuristic Algorithms

1. Introduction

In two recent decades, Cloud Computing (CC) has been emerged as an inevitable paradigm in IT world which awards advantages in many facets such as scalability, elasticity, reliability, robustness, quality of service (QoS), and expenditures reduction to individuals and organizations [1-5]. In this phenomenon cloud providers (e.g. Amazon, Google, and Microsoft) and customers as two major categories of stakeholders interact both economically and technically based on pay-per-use model considering QoS determined in Service Level Agreement (SLA) [1, 6, 7]. CC employs virtualization technology to provide highly scalable and elastic services through Data Centers (DCs) facilities [8]. A DC is a pool of configurable servers, also named Physical Machines

(PMs), which are communicated via a shared network [9, 10]. Customers' requests are delivered to the cloud environment in form of Virtual Machines (VMs) to be placed and processed in PMs. The process of efficiently placing a set of VMs into a minimum number of available PMs to gain a desirable utilization is a well-known NP-Hard problem, called VM Placement (VMP) [11]. Selecting VMP strategy influences variety of DC's benchmark factors chief among energy consumption, computing resources wastage, network performance, and ecological impacts [9].

Numerous studies in literature have focused on VMP with different objectives: mostly focused on power consumption optimization and resources utilization with concern of DCs' cost management and ecological issues [12-28]; and less studied around bandwidth and inter-VMs traffic management to avoid network saturation, reducing performance degradation, and consequently reducing SLA violation rate [29-35]. Although beside the growth of demands for cloud-based services, DCs and communication networks are extended quickly and becoming capable to host and process a greater number of VMs. On the other hand, there are colossal applications from gigantic organizations which are increasingly relying on CC as a daily-routine. Such applications e.g. MapReduce-based applications are often as great as not possible to be handled in form of a single VM. So they are divided into many mutual-communicating VMs to be processed via several PMs over DC's network [35]. Hence an efficient traffic congestion control mechanism is required to cover DCs' tendency for scaling-up in network topology and accepting a greater number of requested VMs with different levels of data affinity. Optimization of network's bandwidth usage in various grades of inter-VMs communications can improve satisfaction level of distributed applications. Co-hosting VMs with high affinity or at least placing them on PMs in a close proximity may reduce traffic stream among VMs on communication links and consequently total performance would be improved for both network and application. In such way, network's depreciation can be decreased which is desirable for provider; and also SLA violation rate can be reduced which is obviously demanded for customer. Some of related works in literature have studied around this issue [32, 33, 35]. But there is a lack for considering impact of different levels of communications on network and applications status.

This paper proposes a combined meta-heuristic communication-aware VMP approach which reduces total traffic stream on network links by placing high affinity VMs in closest neighborhoods as possible, specialized for VL2 topology. Proposed approach performs in three levels of communication patterns: congested, middle, and sparse with comparative point of view. A combination of two well-known Meta-heuristics, Ant Colony Optimization (ACO) and Genetic Algorithm (GA): ACOGA, is employed to improve outcomes in addition to overcome the NP-hard complexity of the problem. Since scaling-up in network's size and hosting incremental numbers of VMs may seriously affect required performance, VL2 [36] topology is considered as case study due to ease of its scalability and possibility of generalizing results for every scale of the network.

The rest of the paper is arranged as follows: an overview on related studies and a brief literature review are presented in sections 2 and 3 respectively. The proposed meta-heuristic approach is described in section 4. Section 5 is dedicated to evaluation of results; and conclusion and future works are represented in section 6.

2. Related Works

In recent years along with growth of demands for cloud-based services, DCs became greater in size and scale. Hence VMP problem has attracted attentions in previous studies with different goals e.g. energy consumption and cost management [37-42]; ecological sustainability [19-26, 39]; and resources utilization [37-39, 41, 43]. Beside emerging colossal applications such as MapReduce based ones, necessity of efficient placing numerous communicating VMs on high scale DCs with the goal of traffic and bandwidth management has been crucial. Previous traffic- and communication-aware VMP researches can be investigated from two aspects: i) VMP strategy; ii) the solution type, which are overviewed in follows.

2.1. Solving approaches

Approaches of solving VMP problem generally fall into 3 classes: i) exact; ii) heuristic; and iii) meta-heuristic. Exact approaches such as Linear Programming (LP) [44-47], and Constraint Programming (CP) [48, 49] are capable to produce precise optimal solution. But due to exponential time complexity of VMP problem (NP-hard) these types of solving approaches are not convenient for real world instances with thousands of VMs and PMs. So they are limited to problems of small sizes and generally are not practical for modern DCs.

Heuristic approaches such as any-Fit-works (First Fit (FF) [50], Best Fit (BF) [51], FF Decreasing (FFD) [52, 53], and pack-works (choose pack and permutation pack [54]) which perform in greedy manner, are capable to generate a single approximation solution in an acceptable time. However heuristics have a high potential to fall into local optimum trap [55, 56]. Therefor the given solution by these class of solving approaches, even with time-superiority against exact approaches, are not promising, specially, for large scale networks accommodating a great number of VMs as in real applicable instances.

The third category of solving approaches, meta-heuristics, most promising among them: Genetic Algorithm (GA) [57], Ant Colony Optimization (ACO) [58], Particle Swarm Optimization (PSO) [59-62], and Simulated Annealing (SA) [63] can solve the VMP problem in a reasonable time by providing approximation solutions near to optimal using randomly generated data. Moreover, since meta-heuristics combine past heuristics to efficiently explore search space, they have a bigger chance to scape local optimal trap in comparison with heuristic algorithms. Although they are approximate in nature and achieving global optimal solution is not guaranteed [56, 29].

2.2. Traffic based VMP strategies

Inter-VM communication is an inevitable challenge which emerged along with DCs' networks scaling-up and growth of requests for high-affinity distributed VMs. This issue may result in network congestion which forms a serious bottleneck for performance for both communication network and applications; and consequently may increase SLA violation rate. Here, previous researches that have focused on traffic and networking issues are overviewed.

In [64] authors presented some policies based on Integer Linear Programming (ILP) and heuristics for fairly allocating bandwidth among communicating VMs considering capacity constraints. They evaluate performance of policies by monitoring total service and waiting times beside service time per unit. Outcomes showed that in cases of small average volumes or low arrival rates results of two heuristics, called Bandwidth Equal Share (BES) and Volume Centric Bandwidth Allocation: Heuristic (VCBA(H)), were close to those given by ILP-based policies. VCBA(H) had the best performance in cases of increasing in volume or arrival rate. Results were achieved from Fat Tree networks with 64 and 256 PMs that for second one ILPs were not tried because of exponential execution time. Also in large size cases of PMs and VMs there is local optimal trap for solution resulted from heuristics.

Authors of [65] presented a Quadratic (QMVMP), two Linear Mathematical models (LMVMP and LMVMP-II), and two heuristic approaches (GRASP and BRKGA) to minimize cost of VMP problem over geographically separated DCs. Small instances of problem were solved using CPLEX general-purpose solver. Computational results showed improvements in lower bounds quality for LMVMP-II. However CPLEX was not practical for larger sizes of problem due to intractable execution time. Then heuristic-based approaches in combination with path-relinking procedure and a local search strategy were evaluated. Results showed a marginally advantage of BRKGA for small instances, while GRASP slightly won the comparisons for larger instances.

A network-aware VMP approach with product traffic pattern (NA-VMP-PT) compatible with Clique or VL2 is proposed in [66], in which an activity level is assigned to each VM that represents the probability of occurring that a random user request may visit such VM. The proposed approach can produce an optimal solution with $n(\log n)$ complexity class in case of homogeneity of PMs, when each of which is able to host a same number of VMs. Although this optimality is not generalizable for all practical settings. Also lack of comparative experiments with other pioneer approaches is tangible.

A Greedy VMP approach with Two Path Routing (GVMTPR) is presented in [67]. The proposed method performs in two phases: firstly it considers the efficiency and impact of VMP on network congestion status when a single shortest routing path is assumed; and secondly it splits the traffic stream and rout them into two separated paths with the goal of guaranteeing available bandwidth in case of failure of a single link or port. Outcomes on 128 servers in Fat Tree topology demonstrated that proposed approach outperformed FFD and random placement approaches. Although inter-VM congestion levels and generalizability of method for larger network scales were not considered in this work.

Authors of [35] presented a 3-objective approach to solve VMP problem with the goals of utilization in computational resources and optimization in energy consumption beside total traffic reduction over DC’s network. Inter-VM communications was arbitrary and known from historical data. They solved the proposed model using a hybrid Multi Objective GA (MOGA) employing an exploitation procedure to perform excessive local search in neighborhoods of achieved solution to improve the chance of finding better results. Outcomes showed that MOGA outperformed Multi Objective ACO (MOACO), FFD, and random placement in a variety of extensive scenarios on Fat Tree and PortLand topologies. Traffic pattern was considered to be same with no variations in all the scenarios.

In [68] a single objective heuristic algorithm is presented to solve VMP problem with the goal of maximizing a metric, called satisfaction, which measures the suitability of a PM for each of its assigned VMs in a heterogeneous network. This work investigated tendency of each VM to receive or process traffic flow via particular computational or connectional nodes of network, which are called sink. Although traffic affinity of VMs was omitted in this greedy-based approach.

A 2-objective meta-heuristic approach is presented in [69] with following goals: reducing the Maximum Link Utilization (MLU) and minimizing traffic over the network. Authors tried to achieve the first objective by balancing VMs during placement process; and co-hosting or at least placing high-affinity VMs under same network switches to achieve second objective. The number of switches routing traffic stream between source and destination PMs was defined as communication cost. A hybrid ACO algorithm enhanced by a 2-opt local search was employed to solve the problem. According to outcomes 37% reduction in the number of hot links and 20% reduction in MLU is reported. Communication among VMs considered to be same with no variations through simulations in this work.

An approach, named VMPlanner, is presented in [32] to reduce power consumption in DC in following three steps: grouping VMs with considering traffic affinity; mapping VMs groups to rack servers with considering physical distance; and routing inter-VMs traffic stream with considering power. In this work the VMP process and traffic stream routing are handled so that much unused network nodes as possible are turned off to save power. Outcomes reported more power saving resulted from VMPlanner in comparison with ElasticTree [70]. The impact of different levels of inter-VM traffic is not met in this approach.

In [29] an ACO-based 3-objective optimization approach is presented for VMP and consolidation problem with the goal of following objectives: reducing energy consumption; reducing resources wastage; and reducing the cost of communication energy. The whole process is divided in two parts: i) an initial VMP on available PMs; ii) consolidating placed VMs using migration to eliminate unwanted event of sprawl servers. Although the affinity among the VMs and its various levels is not investigated in this work.

Authors of [34] presented a VMP approach in 2 phases: i) a link-aware flow threshold-based placement algorithm is proposed; ii) a number of VMP algorithms are offered beside the algorithm of the first phase to improve locality of traffic. The approach is tested on hybrid wireless 60 Giga Hertz links DC networks. Results reported a desired achievement in throughput, time of flow completion, and execution time complexity. Various grades of inter-VM communications are not taken into consideration in this work.

A single objective approach, called MinDistVMDataPlacement, is presented in [71] which performs data and VM placement in a same time, solved by ACO. In this research data are tried to be placed on PMs in close neighborhood. The goal is to optimize traffic over the network and to reduce bandwidth usage by placing a number of VMs that are required. Results showed that presented approach has a better performance against other measure approaches. Nevertheless variety levels of affinity among VMs is not met in this study.

In [72], a model of VMP beside migration is presented to reduce data access time by reducing distance between VMs and their required data which are saved on Storage Attached Networks (SANs). In policy of this research, all the VMs have a uniform access level to needed disk images; and VMs are tried to be placed on servers having more desired condition considering situation between servers and data, with the goal of adhering user SLA. Outcomes indicated that proposed approach has an acceptable performance in data transfer among source and destination VMs.

Authors of [33] presented a traffic-aware VMP Problem (TVMPP) to enhance scaling the network up. The main idea of this study is to place high-affinity VMs in nearest physical hosts as possible. The proposed algorithm is complemented using an approximation algorithm to evaluate it for problems of larger sizes. Results of simulations indicated an implicit dependency between network topology and the advantages of TVMPP. The advantages are desired for multi-layer topologies such as BCube [73], and in contrast they are not so significant for topologies which follow load balancing techniques such as VL2 [36].

2.3. Discussion

Nowadays with increasing tendency of individuals and organizations to cloud-based services, DCs are growing rapidly to accommodate greater numbers of VMs, and specially, large size applications which are divided into many distributed communicating VMs to be processed via several servers. In this regard, selection of an appropriate VMP policy is so essential to handle inter-VM traffic streams in large scale DCs to avoid performance bottlenecks for both network and applications. VMP approaches in related past studies are mostly focused on energy saving, resources utilization, and environmental sustainability; and less investigated around bandwidth usage and VMs' traffic affinities. While network congestion is a key issue which is required to be efficiently solved in modern DCs. Related network- and communication-aware approaches generally tried to co-host high-affinity VMs or at least to place them on PMs in close proximities. However, according to our best knowledge different possible levels of communications among VMs for highly scalable topologies such as VL2 is not studied yet perfectly. Hence, in this work we propose a communication-aware VMP approach by considering inter-VMs traffic patterns in three grades: congested, middle, and sparse, specialized for VL2 topology due to ease of its scaling-up for hosting large size applications.

3. Literature review

3.1. Virtual Machine Placement (VMP)

The process of placing a set of VMs on a number of available PMs, so the hosting PMs are tried to be efficiently utilized, is called VMP (Fig. 1), which belongs to NP-hard problems [9, 79, 80].

In numerous of last studies, VMP has been modeled as a bin-packing problem having dimensions according to constraints implied in problem statement. The goal is to pack VMs as objects in a minimum number of PMs as bins, by considering problem constraints as bin's capacity in each dimension, as shown in Fig. 2. The VMP problems fall into two categories according VMs arrival mode: static (offline), and dynamic (online). In static VMP all of VMs and their requirements are known. So VMs are assigned to available PMs with applying no changes on prior placed VMs. In dynamic mode, network situation is monitored in determined time durations to detect overloaded and underloaded servers. Then some earlier placed VMs may be migrated before assigning new arrived VMs to enhance servers' utilization and to avoid 'server sprawl' phenomenon. In fact, dynamic VMP is a step of consolidation process here which tries to periodically pack existing VMs on smallest number of available PMs, so that idle PMs to be hibernated to save more power [35].

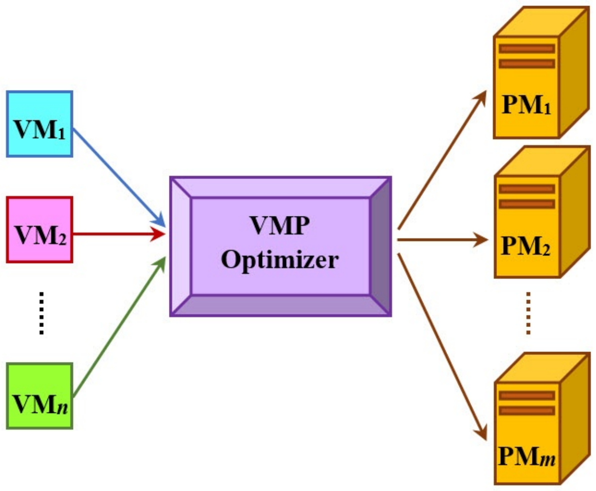


Figure 1. Virtual Machine Placement (VMP) [35].

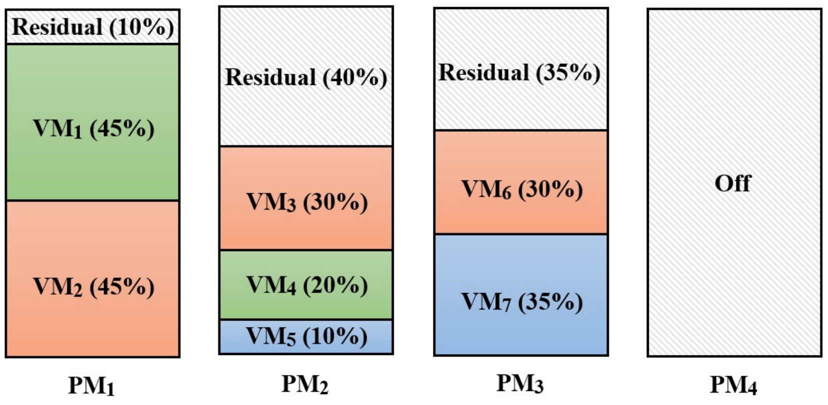


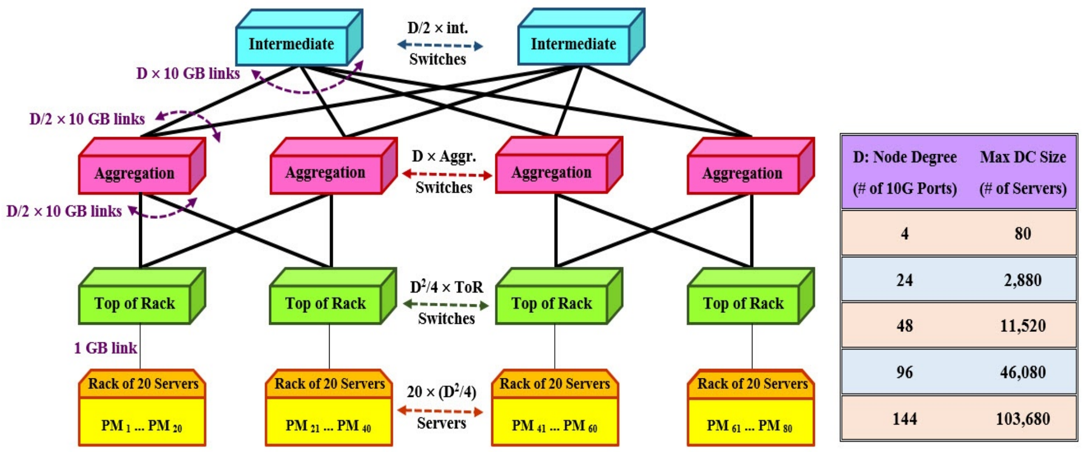
Figure 2. One dimensional bin-packing schema [9].

3.2. VL2 Topology

The term 'topology' is used for two concepts in cloud DCs' literature: i) the pattern of communication among VMs, and ii) the DC's shared network architecture which interconnects PMs [74]. Common DC network topologies are generally classified as following [75]:

i. Hierarchical models such as Fat-Tree [74], Portland [72], and VL2 [36]
ii. Resource models such as Dcell [76] and Bcube [73]
iii. Rack-to-rack models such as Scafida [77] and Jelly fish [78]

VL2 is a scalable flexible three tier hierarchical model which scales-up by increasing the number of ports of switches (node degree) in different levels as shown in Fig. 3. As shown in Fig. 3, the highest level, core layer, contains $D/2$ intermediate switches which are connected to D aggregation switches via 10 GB links. Top of Rack (ToR) switches in lowest level, access layer, interconnect their underlying racks, each of which containing 20 to 40 PMs, to the aggregation switches. Traffic is first routed from PMs to corresponding ToR switch using 1 GB link and then forwarded to a randomly selected intermediate switch in core layer via aggregation switch. Next the traffic is forwarded back to its real destination in access layer [36].



D: Node Degree (# of 10G Ports)	Max DC Size (# of Servers)
4	80
24	2,880
48	11,520
96	46,080
144	103,680

Figure 3. VL2 topology [36].

4. Problem definition

This study is concentrated on optimizing overall traffic stream among VMs specialized for scalable DC network topology, VL2. In this regard, inter-VM Communication Matrix (CM), algorithm of calculating traffic transferred between each arbitrary pair of VMs, and optimization function for overall network traffic are defined in this section.

Various VMs, specially, in distributed applications may have dependencies to be completely executed. The type, direction and volume of traffic between VMs can be extracted from historical data as VMs' expected behaviors. Traffic dependencies among VMs are shown in form of a matrix as defined below:

$$CM = \begin{array}{c|cccc} & VM_1 & VM_2 & \cdots & VM_n \\ \hline VM_1 & 0 & 30 & \cdots & 800 \\ VM_2 & 670 & 0 & \cdots & 140 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ VM_n & 580 & 10 & \cdots & 0 \end{array} \qquad (1)$$

In VMP each VM can be assigned to each arbitrary PM. The communication between two VMs directly depends on the physical distance between their host PMs. The physical distance between each pair of PMs can be calculated by counting the number of 'hops' between them. A hop is a step from one node of the network to the other one. In VL2 topology, the distance between each pair of PMs can be calculated according to Algorithm 1. Generally in VL2 topology each rack underpins 20 to 40 servers. The VMs hosted by a single PM have no communication cost. A pair of dependent VMs hosted by PMs under a ToR switch have to transfer traffic to each other by passing two hops. In other situations, communication traffic passes six hops to reach destination.

This study formulates an approach aim to optimize total traffic stream over network by placing dependent VMs on PMs in proximities as close as possible. In this manner, the number of hops between dependent VMs is reduced, so consequently the total traffic flow on the network is reduced.

By considering distance of PM_k and PM_ℓ hosting VM_i and VM_j respectively, the Total Traffic Flow (TTF) over the network is defined as equation (2):

$$TTF = \min \sum_{j=1}^{n} Distance(k, \ell) \times CM(i, j) \tag{2}$$
$$\forall \; k, \ell \in \{1, 2, \ldots, m\}$$
$$\forall \; i \in \{1, 2, \ldots, n\}$$

Where n is the number of VMs and m is the number of PMs. To solve defined optimization problem, a combination of ACO and GA, named ACOGA, is employed which is explained in detail in next section.

Algorithm 1. Distance

Input: k, ℓ (index of PMs hosting VM_i and VM_j respectively), r (number of PMs in a rack), ε (very small positive number)
Output: distance of PM_k and PM_ℓ

```
1: Begin
2:   if  k = ℓ  then  distance = 0;  % --- distance inside a same PM
3:   else if  $\lfloor \frac{k-\varepsilon}{r} \rfloor = \lfloor \frac{\ell-\varepsilon}{r} \rfloor$  then  distance = 2;  % --- distance under a ToR switch
4:   else distance = 6;  % --- distance out of ToR switch
5:   return distance;
6: End % --- end of Algorithm 1. Distance
```

5. Proposed communication-aware approach

To solve the communication-aware VMP problem defined in former section, this study represents a combined meta-heuristic approach based on co-working two well-known meta-heuristic algorithms: ACO and GA, named ACOGA. A brief chart of ACOGA is illustrated in Fig. 4. We can enumerate merits and demerits of both ACO and GA. Then, we conduct mixture of ACOGA in such a way to engage only their plus points. ACO converges quickly and GA has good exploration ability, but it is a little slowly. Similar, to other meta-heuristic algorithms GA starts with initial population. To speedup this algorithm, we exploit ACO to generate semi-random, but sub-optimal population which constructs part of whole initial population. The rest population is purely generated randomly. This experiment let to promising results in which ACOGA beats both GA and ACO individually. One important thing to mention is that these two algorithms are flexible to be customized for discrete VMP problem in contrast with new meta-heuristic algorithms such as Grey-wolf and even Whale optimization algorithm that are continuous in nature.

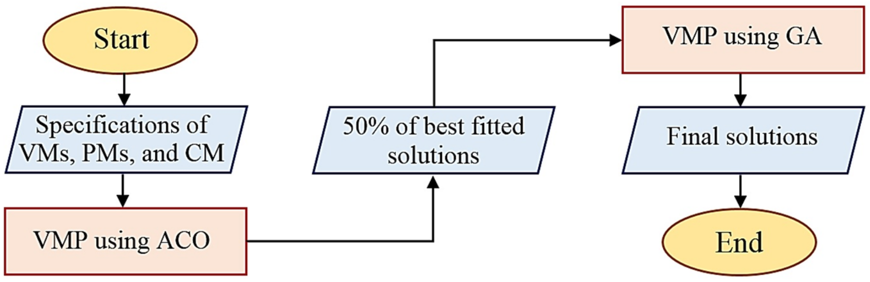

Start
Specifications of VMs, PMs, and CM
VMP using ACO
50% of best fitted solutions
VMP using GA
Final solutions
End

Figure 4. Chart of proposed ACOGA

As shown in Fig. 4, firstly the problem is solved using ACO as usual. In the next step, the problem is solved using GA which is fed by 50% of the best solutions produced by the ACO in the arrangement of initial population. High level pseudo codes of ACO and GA are implied in Algorithm 2 and Algorithm 3 respectively.

An initial random solution is created by placing n VMs on m PMs, so that each PM can accommodate at most 4 VMs up to its capacities thresholds. VMs by considering their requirements for memory and processor as problem constraints are placed on PMs in form of 2-dimenssional bin packing problem with the goal of minimizing TTF defined in equation (2).

In both algorithms, 2 and 3, the fitness function calculates the TTF of each solution. Obviously smaller values of TTF are desirable. Therefor a solution with the smallest fitness value is the best VMP result. However, beside TTF reduction, the number of PMs in use and idle PMs are monitored to save energy and resources.

All the steps of ACOGA are applied for three CM modes separately:

- Sparse: less than 25% of VMs have communication
- Middle: between 25% and 75% of VMs have communication
- Congested: more than 75% of VMs have communication

Algorithm 2. VMP by ACO

Input: VMs requirements, PMs capacities, CM, ACO parameters

Output: a set of VMP solutions

```
1: Begin
2:  hop = distance (each pair of PMs);
3:  % --- initial solutions
4:  for s = 1 : number of ants
5:      % --- a VMP arrangement as initial solutions[s]
6:      for i = 1 : number of PMs
7:          for j = 1 : number of VMs
8:              for k = 1 : maximum number of VMs allowed for each PM up to capacities
9:                  if VM[j]'s requirements ≤ PM[i]'s capacities
10:                     then place VM[j] on PM[i][k]
11:                 end
12:             end
13:         end
14:     end
15:     % --- initial solutions[s] is created
16: end
17:  solutions fitness = fitness (initial solutions);
18:  pheromone = update (pheromone);  % --- based on initial solutions fitness
19:  for iteration = 1: maximum iteration
```

```
20:         new solutions = next solutions (pheromone);
21:         new solutions fitness = fitness (new solutions);
22:         new pheromone = update (pheromone);  % --- based on new solutions fitness
23:         idle servers = number of idle PMs in solutions[iteration];
24:         solutions = new solutions;
25:         pheromone = new pheromone;
26:   end
27:   best ACO solutions = 50 percent of most fitted solutions;  % --- will be used to feed GA
28: End % --- end of Algorithm 2. VMP by ACO
```

Next section is dedicated to comparison and evaluation of proposed ACOGA against other state-of-the-art rival approaches.

Algorithm 3. VMP by GA

Input: VMs requirements, PMs capacities, CM, GA parameters, best ACO solutions
Output: a set of VMP solutions

```
1: Begin
2:   hop = distance (each pair of PMs);
3:   % --- half of initial population
4:   for p = 1 : ⌊population size/2⌋
5:     % --- a VMP arrangement as initial population[p]
6:     for i = 1 : number of PMs
7:       for j = 1 : number of VMs
8:         for k = 1 : maximum number of VMs allowed for each PM up to capacities
9:           if VM[j]'s requirements ≤ PM[i]'s capacities
10:            then place VM[j] on PM[i][k]
11:          end
12:        end
13:      end
14:    end
15:    % --- initial population[p] (chromosome (p)) is created
16: end
17: % --- best ACO solutions are inserted as 2nd half of initial population
18:  for p = ⌊population size/2⌋ + 1 : population size
19:    Initial population[(⌊population size/2⌋ + 1) .. population size] ← best ACO solutions
20:  end
21:  for iteration = 1: maximum iteration
22:    population fitness = fitness (population);  % --- based on initial population  fitness
23:    idle servers = number of idle PMs in population[iteration];
24:    parents = roulette wheel (population);
25:    children = crossover (parents);
26:    correct children = check and correct (children);  % --- correction of chromosomes
27:    muted = mutation (population);
28:    intermediate population = concatenate (correct children, muted, elites of population);
29:    for p = 1: population size
30:      new population [p] = random selection (intermediate population);
31:    end
32:    population = new population;
33:  end for
34:  return population[p-1];  % --- returns population of final iteration
35: End % --- end of Algorithm 3. VMP by GA
```

6. Results and evaluations

6.1. Parameters, settings, and simulations

As mentioned earlier, this research studies TTF specialized for VL2 DC topology in three levels: sparse, middle, and congested using ACOGA. The proposed approach is applied for 80 homogeneous PMs, 20 PMs per rack, and 120 VMs. Homogeneous PMs are of same configuration. In this study CPU and Memory capacities of PMs are considered 6000 MIPS and 8 GB respectively. However, for heterogeneous servers with different configurations or in case of existing probable bandwidth limitations, our current assumptions require some changes to adopt new situation.

For evaluating the results, a comparative analysis is represented in this section against a heuristic greedy based approach; and two meta-heuristics: GA, and ACO. GA and ACO are solved as usual for data and specifications of the problem of this study with parameters and settings as depicted in Table 1.

Table 1. Parameters and settings

Approach	**GA**	**ACO**
Parameters and Settings	Population size: 100 Maximum iteration: 50 Crossover rate: 0.8 Mutation rate: 0.1	Number of ants: 100 Maximum iteration: 50 Alpha: 0.8 Beta: 0.2 Evaporation rate: 0.1 Minimum level of pheromone: 0.01 Maximum level of pheromone: 1.99

For greedy approach, firstly VMs communications in CM are sorted descending, and then VMs with highest volume of traffic transfer are tried to be co-hosted or placed in nearest proximities. Because of randomness nature of meta-heuristic approaches, the simulations are repeated 20 times for each meta-heuristic approach and the average values are considered as result. Outcomes are depicted in Fig. 5 and Table 2. As Fig. 5 shows, three meta-heuristic approaches have a significant superiority against heuristic greedy one in all the CM modes. Also proposed ACOGA outperformed two near rival approaches, ACO and GA, specially, in congested mode. Fig. 6 and also Table 2 show the average of idle servers. The numbers of idle PMs in different CM modes imply advantage of proposed ACOGA from energy and resource saving point of views.

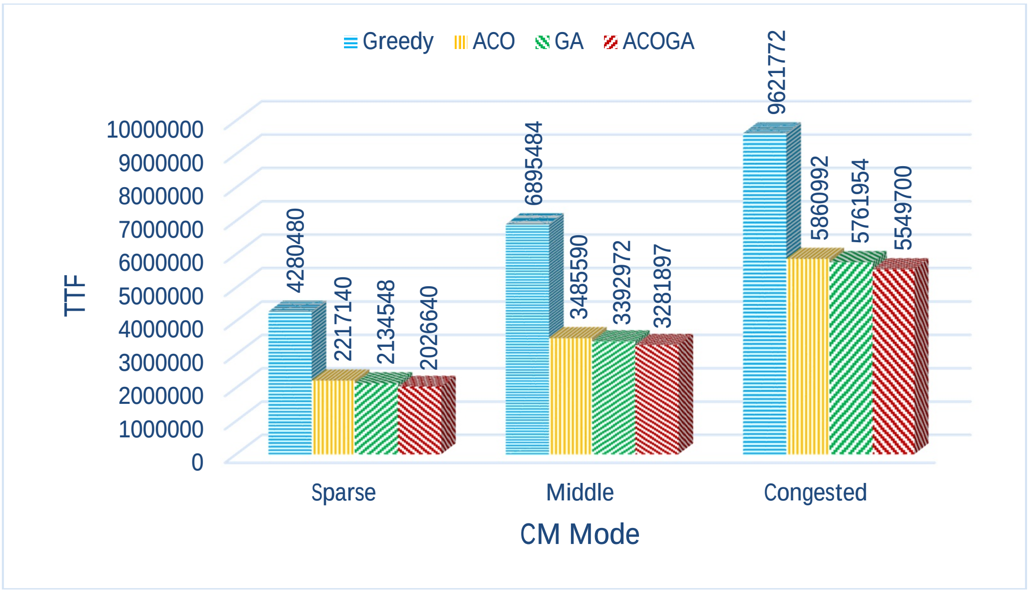


Figure 5. Comparison and evaluation of results

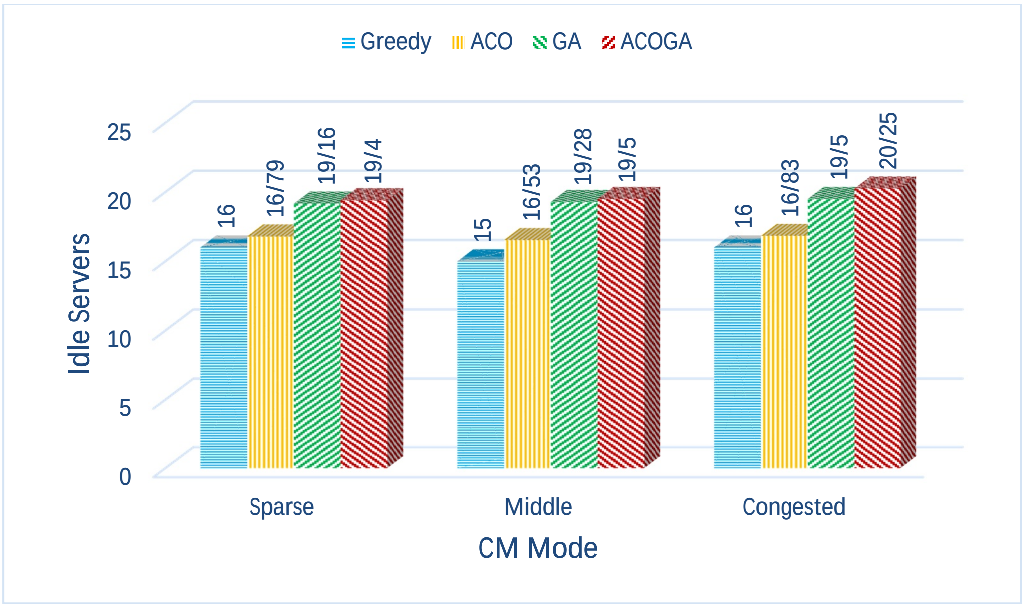


Figure 6. Average of idle servers

Table 2. Simulations results

	Sparse		Middle		Congested	
	TTF	Idle PMs	TTF	Idle PMs	TTF	Idle PMs
Greedy	4280480	16	6895484	15	9621772	16
ACO	2217140	16.79	3485590	16.53	5860992	16.83
GA	2134548	19.16	3392972	19.28	5761954	19.5
ACOGA	**2026640**	**19.4**	**3281897**	**19.5**	**5549700**	**20.25**

6.2. Time complexity

The proposed ACOGA approach is a sequential execution of two meta-heuristics: GA and ACO. Therefore, the overall time complexity can be achieved by calculating the complexity of two mentioned algorithms separately. There are many procedures in the body of ACO and GA, which consist ACOGA together. So, the complexity of each meta-heuristics is reached by calculating the total complexity of procedures. In this regard, having n VMs, m PMs, by considering the parameters defined in Table 1, and the procedures shown in Algorithms 1-3, we have time complexities as depicted in Table 3. As shown in Table 3, the number of solutions, namely number of ants and population size, in both meta-heuristics are determined by the user in initializing phase as fixed values. Also crossover rate is usually defined as a positive value less than 1, which can be ignored as coefficient of population size. Since these two meta-heuristics are sequentially executed, the total number of solutions can be omitted as a fixed coefficient in overall runtime. Therefore, the time complexity of ACOGA equals:

$$O[(\text{number of ants} + \text{crossover rate} \times \text{population size}) \times (\text{maximum iteration} \times m^2 \times n^2)] =$$
$$O(\text{maximumiteration} \times m^2 \times n^2)$$

Table 3. Time complexity

	Procedure	Complexity
ACO	Distance	$O(1)$
	Solution	$O(\text{number of ants} \times m \times n^2)$
	Fitness	$O(\text{number of ants} \times m^2 \times n^2)$
	Update	$O(\text{number of ants} \times m \times n)$
	Idle servers	$O(\text{number of ants} \times m \times n)$
Overall for ACO: $O(\text{maximum iteration} \times \text{number of ants} \times m^2 \times n^2)$		
	Procedure	Complexity
GA	Distance	$O(1)$
	Population	$O(\text{population size} \times m \times n^2)$
	Fitness	$O(\text{population size} \times m^2 \times n^2)$
	Idle servers	$O(\text{population size} \times m \times n)$
	Roulette wheel	$O(\text{population size}^2 \times \text{crossover rate})$
	Crossover	$O(\text{population size} \times \text{crossover rate})$
	Check and correct	$O(\text{population size} \times \text{crossover rate} \times m^2 \times n^2)$
	Mutation	$O(\text{population size} \times \text{mutation rate})$
	Random selection	$O(\text{population size})$
Overall for GA: $O(\text{maximum iteration} \times \text{population size} \times \text{crossover rate} \times m^2 \times n^2)$		

Complexity of ACOGA:

$$\begin{aligned}
&O(\text{maximum iteration} \times \text{number of ants} \times m^2 \times n^2) + \\
&O(\text{maximum iteration} \times \text{population size} \times \text{crossover rate} \times m^2 \times n^2) = \\
&O[(\text{number of ants} + \text{crossover rate} \times \text{population size}) \times \\
&\quad (\text{maximum iteration} \times m^2 \times n^2)] = \\
&\qquad\qquad \mathbf{O(\text{maximum iteration} \times m^2 \times n^2)}
\end{aligned}$$

7. Conclusion

Cloud computing plays an inevitable role in today’s IT world. Beside the growth of demands for cloud based services, great and distributed applications are relying on cloud environment as a daily routine. Great and distributed applications often require to be dispersed over many dependent virtual machines with mutual communications to be processed as an atomic execution unit. Each virtual machine can be placed on an arbitrary physical machine in a data center. Communications among virtual machines may saturate network links, threaten performance of both data center and application, and consequently increase communication energy consumption and service level agreement violation rate. This paper addressed a combined communication-aware meta-heuristic based approach, named ACOGA, to optimize total traffic flow (TTF) on the network along with reducing active servers to save energy, and utilize resources efficiently. The proposed ACOGA performs TTF optimization by co-hosting high affinity virtual machines or placing them on physical machines in close proximities, specialized for scalable flexible VL2 data center network topology. Simulation results proved the superiority of proposed ACOGA against greedy, ACO, and GA approaches in the sense of communication reduction beside energy and resources saving. For future directions, TTF for heterogeneous PMs or limited bandwidth access can be studied. Also optimization of inter-VMs traffic stream for concatenation of two or more different topologies seems attractive to follow.

References

[1] Hosseini Shirvani, M., Rahmani, A. M., & Sahafi, A. (2018). An iterative mathematical decision model for cloud migration: A cost and security risk approach. Software: Practice and Experience, 48(3), 449-485. https://doi.org/10.1002/spe.2528.

[2] Hosseini Shirvani, M., Amirsoleimani N., Salimpour S., & Azab A.(2017, May). Multicriteria task scheduling in distributed systems based on Fuzzy TOPSIS, in: Presented at 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, Canada, 30 April–3 May, 2017.

[3] Randles, M., Lamb, D., Odat, E., & Taleb-Bendiab, A. (2011). Distributed redundancy and robustness in complex systems. Journal of Computer and System Sciences, 77(2), 293-304. https://doi.org/10.1016/j.jcss.2010.01.008.

[4] Hosseini Shirvani, M. (2018, July). Web Service Composition in multi-cloud environment: A bi-objective genetic optimization algorithm. In 2018 Innovations in Intelligent Systems and Applications (INISTA) (pp. 1-6). IEEE. https://doi.org/10.1109/INISTA.2018.8466267.

[5] Hosseini Shirvani, M. and Babazadeh Gorji, A. (2020). Optimisation of automatic web services composition using genetic algorithm, Int. J. Cloud Computing, 9(2), 1–15.

[6] Pires, F. L., & Barán, B. (2013, December). Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (pp. 203-210). IEEE Computer Society. https://doi.org/10.1109/UCC.2013.44.

[7] Hosseini Shirvani, M. (2020). To move or not to move: An iterative four-phase cloud adoption decision model for IT outsourcing based on TCO. J. Soft Comput. Inf. Technol. Vol 9(2), pp:7-17.

[8] Hosseini Shirvani, M. (2020). A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. Engineering Application of Artificial Intelligence, 90, 1-20.

[9] Hosseini Shirvani, M., Rahmani, A. M., & Sahafi, A. (2020). A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges. Journal of King Saud University-Computer and Information Sciences, 32(3), 267-286. https://doi.org/10.1016/j.jksuci.2018.07.001.

[10] Hosseini Shirvani, M. and Ghojoghi, A. (2018). Server Consolidation Schemes in Cloud Computing Environment: A Review. European Journal of Engineering Research and Science. 1, 3 (Jul. 2018), 18-24.

[11] Bilal, K., Khan, S. U., Kolodziej, J., Zhang, L., Hayat, K., Madani, S. A., ... & Chen, D. (2012, May). A Comparative Study Of Data Center Network Architectures. In ECMS (pp. 526-532). http://dx.doi.org/10.7148/2012-0526-0532.

[12] Krzywda, J., Ali-Eldin, A., Carlson, T. E., Östberg, P. O., & Elmroth, E. (2018). Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling. Future Generation Computer Systems, 81, 114-128. https://doi.org/10.1016/j.future.2017.10.044.

[13] Rizvandi, N. B., Taheri, J., & Zomaya, A. Y. (2011). Some observations on optimal frequency selection in DVFS-based energy consumption minimization. Journal of Parallel and Distributed Computing, 71(8), 1154-1164. https://doi.org/10.1016/j.jpdc.2011.01.004.

[14] Stavrinides, G. L., & Karatza, H. D. (2019). An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. Future Generation Computer Systems, 96, 216-226. https://doi.org/10.1016/j.future.2019.02.019.

[15] Safari, M., & Khorsand, R. (2018). Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. Simulation Modelling Practice and Theory, 87, 311-326. https://doi.org/10.1016/j.simpat.2018.07.006.

[16] Wu, T., Gu, H., Zhou, J., Wei, T., Liu, X., & Chen, M. (2018). Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud. Journal of Systems Architecture, 84, 12-27. https://doi.org/10.1016/j.sysarc.2018.03.001.

[17] Wu, C. M., Chang, R. S., & Chan, H. Y. (2014). A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Generation Computer Systems, 37, 141-147. https://doi.org/10.1016/j.future.2013.06.009.

[18] Guérout, T., Monteil, T., Da Costa, G., Calheiros, R. N., Buyya, R., & Alexandru, M. (2013). Energy-aware simulation with DVFS. Simulation Modelling Practice and Theory, 39, 76-91. https://doi.org/10.1016/j.simpat.2013.04.007.

[19] De Courchelle, I., Guérout, T., Da Costa, G., Monteil, T., & Labit, Y. (2019). Green energy efficient scheduling management. Simulation Modelling Practice and Theory, 93, 208-232. https://doi.org/10.1016/j.simpat.2018.09.011.

[20] Ilic, M. D. (2010). Dynamic monitoring and decision systems for enabling sustainable energy services. Proceedings of the IEEE, 99(1), 58-79. https://doi.org/10.1109/JPROC.2010.2089478.

[21] Garg, S. K., Yeo, C. S., & Buyya, R. (2011, August). Green cloud framework for improving carbon efficiency of clouds. In European Conference on Parallel Processing (pp. 491-502). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23400-2_45.

[22] Bhattacherjee, S., Das, R., Khatua, S., & Roy, S. (2019). Energy-efficient migration techniques for cloud environment: a step toward green computing. The Journal of Supercomputing, 1-29. https://doi.org/10.1007/s11227-019-02801-0.

[23] Khosravi, A., Garg, S. K., & Buyya, R. (2013, August). Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In European Conference on Parallel Processing (pp. 317-328). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40047-6_33.

[24] Usman, M. J., Ismail, A. S., Chizari, H., Abdul-Salaam, G., Usman, A. M., Gital, A. Y., ... & Aliyu, A. (2019). Energy-efficient Virtual Machine Allocation Technique Using Flower Pollination Algorithm in Cloud Datacenter: A Panacea to Green Computing. Journal of Bionic Engineering, 16(2), 354-366. https://doi.org/10.1007/s42235-019-0030-7.

[25] Jing, S. Y., Ali, S., She, K., & Zhong, Y. (2013). State-of-the-art research study for green cloud computing. The Journal of Supercomputing, 65(1), 445-468. https://doi.org/10.1007/s11227-011-0722-1.

[26] Gu, C., Fan, L., Wu, W., Huang, H., & Jia, X. (2018). Greening cloud data centers in an economical way by energy trading with power grid. Future Generation Computer Systems, 78, 89-101. https://doi.org/10.1016/j.future.2016.12.029.

[27] Hosseinzadeh, S., Hosseini Shirvani, M. (2015). Optimizing energy consumption in clouds by using genetic algorithm. Journal of multidisciplinary engineering science and technology, 2(6),pp: 1431-1434.

[28] Mokaripoor, P., Hosseini Shirvani, M. (2016). A state of the art survey on DVFS techniques in Cloud Computing Environment. J. Multidiscip. Eng. Sci. Technol.3 (5).

[29] Malekloo, M. H., Kara, N., & El Barachi, M. (2018). An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments. Sustainable Computing: Informatics and Systems, 17, 9-24. https://doi.org/10.1016/j.suscom.2018.02.001.

[30] Ilkhechi, A. R., Korpeoglu, I., & Ulusoy, Ö. (2015). Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components. Computer Networks, 91, 508-527. https://doi.org/10.1016/j.comnet.2015.08.042.

[31] Shabeera, T. P., Kumar, S. M., Salam, S. M., & Krishnan, K. M. (2017). Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm. Engineering Science and Technology, an International Journal, 20(2), 616-628. https://doi.org/10.1016/j.jestch.2016.11.006.

[32] Fang, W., Liang, X., Li, S., Chiaraviglio, L., & Xiong, N. (2013). VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. Computer Networks, 57(1), 179-196. https://doi.org/10.1016/j.comnet.2012.09.008.

[33] Meng, X., Pappas, V., & Zhang, L. (2010, March). Improving the scalability of data center networks with traffic-aware virtual machine placement. In 2010 Proceedings IEEE INFOCOM (pp. 1-9). IEEE. www.cmlab.csie.ntu.edu.tw/~freetempo/files/Traffic-aware_VM_Placement.pdf.

[34] Roh, H., Jung, C., Kim, K., Pack, S., & Lee, W. (2017). Joint flow and virtual machine placement in hybrid cloud data centers. Journal of Network and Computer Applications, 85, 4-13. https://doi.org/10.1016/j.jnca.2016.12.006.

[35] Farzai S, Hosseini Shirvani, M., Rabbani M. (2020). Multi-Objective Communication-Aware Optimization for Virtual Machine Placement in Cloud Datacenters, Sustainable Computing: Informatics and Systems (2020), doi: https://doi.org/10.1016/j.suscom.2020.100374.

[36] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., . & Sengupta, S. (2009, August). VL2: a scalable and flexible data center network. In ACM SIGCOMM computer communication review (Vol. 39, No. 4, pp. 51-62). ACM. https://doi.org/10.1145/1592568.1592576.

[37] Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79(8), 1230-1242. https://doi.org/10.1016/j.jcss.2013.02.004.

[38] Gopu, A., & Venkataraman, N. (2015). Optimal VM placement in distributed cloud environment using MOEA/D. Soft Computing, 1-20. https://doi.org/10.1007/s00500-018-03686-6.

[39] Xu, J., & Fortes, J. A. (2010, December). Multi-objective virtual machine placement in virtualized data center environments. In 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (pp. 179-188). IEEE. https://doi.org/10.1109/GreenCom-CPSCom.2010.137.

[40] Xiao, Z., Jiang, J., Zhu, Y., Ming, Z., Zhong, S., & Cai, S. (2015). A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory. Journal of Systems and Software, 101, 260-272. https://doi.org/10.1016/j.jss.2014.12.030.

[41] Tordsson, J., Montero, R. S., Moreno-Vozmediano, R., & Llorente, I. M. (2012). Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future generation computer systems, 28(2), 358-367. https://doi.org/10.1016/j.future.2011.07.003.

[42] Mishra, S. K., Puthal, D., Sahoo, B., Jayaraman, P. P., Jun, S., Zomaya, A. Y., & Ranjan, R. (2018). Energy-efficient VM-placement in cloud data center. Sustainable computing: informatics and systems, 20, 48-55. https://doi.org/10.1016/j.suscom.2018.01.002.

[43] Shaw, R., Howley, E., & Barrett, E. (2019). An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. Simulation Modelling Practice and Theory, 93, 322-342. https://doi.org/10.1016/j.simpat.2018.09.019.

[44] Lin, J. W., Chen, C. H., & Lin, C. Y. (2014). Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications. Future Generation Computer Systems, 37, 478-487. https://doi.org/10.1016/j.future.2013.12.034.

[45] Chaisiri, S., Lee, B. S., & Niyato, D. (2009, December). Optimal virtual machine placement across multiple cloud providers. In 2009 IEEE Asia-Pacific Services Computing Conference (APSCC) (pp. 103-110). IEEE. https://doi.org/10.1109/APSCC.2009.5394134.

[46] Bichler, M., Setzer, T., & Speitkamp, B. (2006). Capacity planning for virtualized servers. In Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA. https://ssrn.com/abstract=1025862.

[47] Speitkamp, B., & Bichler, M. (2010). A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Transactions on services computing, 3(4), 266-278. https://doi.org/10.1109/TSC.2010.25.

[48] Van, H. N., Tran, F. D., & Menaud, J. M. (2010, July). Performance and power management for cloud infrastructures. In 2010 IEEE 3rd international Conference on Cloud Computing (pp. 329-336). IEEE. https://doi.org/10.1109/CLOUD.2010.25.

[49] Hermenier, F., Lorca, X., Menaud, J. M., Muller, G., & Lawall, J. (2009, March). Entropy: a consolidation manager for clusters. In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (pp. 41-50). ACM. https://doi.org/10.1145/1508293.1508300.

[50] Yue, M. (1991). A simple proof of the inequality FFD (L)≤ 11/9 OPT (L)+ 1,∀ L for the FFD bin-packing algorithm. Acta mathematicae applicatae sinica, 7(4), 321-331. https://doi.org/10.1007/BF02009683.

[51] Grit, L., Irwin, D., Yumerefendi, A., & Chase, J. (2006, November). Virtual machine hosting for networked clusters: Building the foundations for autonomic orchestration. In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (p. 7). IEEE Computer Society. https://doi.org/10.1109/VTDC.2006.17.

[52] Baker, B. S. (1985). A new proof for the first-fit decreasing bin-packing algorithm. Journal of Algorithms, 6(1), 49-70. https://doi.org/10.1016/0196-6774(85)90018-5.

[53] Kao, M. Y. (Ed.). (2008). Encyclopedia of algorithms. Springer Science & Business Media. ISBN: 978-0-387-30162-4.

[54] Leinberger, W., Karypis, G., & Kumar, V. (1999, September). Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In Proceedings of the 1999 International Conference on Parallel Processing (pp. 404-412). IEEE. https://doi.org/10.1109/ICPP.1999.797428.

[55] Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. Handbook of combinatorial optimization, 455-531. https://doi.org/10.1007/978-1-4419-7997-1_35.

[56] Festa, P. (2014, July). A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In 2014 16th International Conference on Transparent Optical Networks (ICTON) (pp. 1-20). IEEE. https://doi.org/10.1109/ICTON.2014.6876285.

[57] Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., & Yuan, L. (2010, July). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In 2010 IEEE International Conference on Services Computing (pp. 514-521). IEEE. https://doi.org/10.1109/SCC.2010.69.

[58] Feller, E., Rilling, L., & Morin, C. (2011, September). Energy-aware ant colony based workload placement in clouds. In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing (pp. 26-33). IEEE Computer Society. https://doi.org/10.1109/Grid.2011.13.

[59] Jeyarani, R., Nagaveni, N., & Ram, R. V. (2013). Self adaptive particle swarm optimization for efficient virtual machine provisioning in cloud. In Organizational Efficiency through Intelligent Information Technologies (pp. 88-107). IGI Global. https://doi.org/10.4018/978-1-4666-2047-6.ch006.

[60] Jeyarani, R., Nagaveni, N., & Ram, R. V. (2012). Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. Future Generation Computer Systems, 28(5), 811-821. https://doi.org/10.1016/j.future.2011.06.002.

[61] Hosseini Shirvani, M. (2020). Bi-objective web service composition problem in multi-cloud environment: a bi-objective time-varying particle swarm optimisation algorithm, Journal of Experimental & Theoretical Artificial Intelligence, DOI:10.1080/0952813X.2020.1725652.

[62] Lotfi, H., Ghazi, R., Naghibi Sistani, M. (2019). An Improved Particle Swarm Optimization Algorithm for Energy Management in Distribution Grid Considering Distributed Generators. Journal of Advances in Computer Research, 10(3), 1-10.

[63] Liao, X., Jin, H., & Liu, H. (2012). Towards a green cluster through dynamic remapping of virtual machines. Future Generation Computer Systems, 28(2), 469-477. https://doi.org/10.1016/j.future.2011.04.013.

[64] Karmakar, K., Das, R. K., & Khatua, S. (2020). Bandwidth allocation for communicating virtual machines in cloud data centers. The Journal of Supercomputing, 1-22. https://doi.org/10.1007/s11227-019-03128-6

[65] Stefanello, F., Aggarwal, V., Buriol, L. S., & Resende, M. G. (2019). Hybrid algorithms for placement of virtual machines across geo-separated data centers. Journal of Combinatorial Optimization, 38(3), 748-793. https://doi.org/10.1007/s10878-019-00411-3

[66] You, K., Tang, B., & Ding, F. (2013, June). Near-optimal virtual machine placement with product traffic pattern in data centers. In 2013 IEEE International Conference on Communications (ICC) (pp. 3705-3709). IEEE. https://doi.org/10.1109/ICC.2013.6655130

[67] Kanagavelu, R., Lee, B. S., Le, N. T. D., Mingjie, L. N., & Aung, K. M. M. (2014). Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. Computer Communications, 53, 1-12. https://doi.org/10.1016/j.comcom.2014.07.009

[68] Ilkhechi, A. R., Korpeoglu, I., & Ulusoy, Ö. (2015). Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components. Computer Networks, 91, 508-527. https://doi.org/10.1016/j.comnet.2015.08.042.

[69] DONG, J. K., WANG, H. B., LI, Y. Y., & CHENG, S. D. (2014). Virtual machine placement optimizing to improve network performance in cloud data centers. The Journal of China Universities of Posts and Telecommunications, 21(3), 62-70. https://doi.org/10.1016/S1005-8885(14)60302-2.

[70] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., & McKeown, N. (2010, April). Elastictree: Saving energy in data center networks. In Nsdi (Vol. 10, pp. 249-264). https://www.usenix.org/legacy/event/nsdi10/tech/full_papers/heller.pdf.

[71] Shabeera, T. P., Kumar, S. M., Salam, S. M., & Krishnan, K. M. (2017). Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm. Engineering Science and Technology, an International Journal, 20(2), 616-628. https://doi.org/10.1016/j.jestch.2016.11.006.

[72] Piao, J. T., & Yan, J. (2010, November). A network-aware virtual machine placement and migration approach in cloud computing. In 2010 Ninth International Conference on Grid and Cloud Computing (pp. 87-92). IEEE. https://doi.org/10.1109/GCC.2010.29

[73] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., ... & Lu, S. (2009). BCube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review, 39(4), 63-74. https://doi.org/10.1145/1594977.1592577.

[74] Cao, G. (2019). Topology-aware multi-objective virtual machine dynamic consolidation for cloud datacenter. Sustainable Computing: Informatics and Systems, 21, 179-188. https://doi.org/10.1016/j.suscom.2019.01.004.

[75] Masdari, M., Gharehpasha, S., Ghobaei-Arani, M., & Ghasemi, V. (2019). Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. Cluster Computing, 1-31. https://doi.org/10.1007/s10586-019-03026-9

[76] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., & Lu, S. (2008, August). Dcell: a scalable and fault-tolerant network structure for data centers. In Proceedings of the ACM SIGCOMM 2008 conference on Data communication (pp. 75-86). https://doi.org/10.1145/1402958.1402968

[77] Gyarmati, L., & Trinh, T. A. (2010). Scafida: A scale-free network inspired data center architecture. ACM SIGCOMM Computer Communication Review, 40(5), 4-12. https://doi.org/10.1145/1880153.1880155

[78] Singla, A., Hong, C. Y., Popa, L., & Godfrey, P. B. (2012). Jellyfish: Networking data centers randomly. In Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12) (pp. 225-238).

[79] Hosseini Shirvani, M. S. (2018). A new Shuffled Genetic-based Task Scheduling Algorithm in Heterogeneous Distributed Systems. Journal of Advances in Computer Research, 9(4), 19-36. http://jacr.iausari.ac.ir/article_660143.html.

[80] Hosseini Shirvani, M. (2015). Evaluating of Feasible Solutions on Parallel Scheduling Tasks with DEA Decision Maker. Journal of Advances in Computer Research, 6(2), 109-115. http://jacr.iausari.ac.ir/article_641726.html.