Virtual Machine Consolidation in Cloud Data Centers using ACO Metaheuristic

Md Hasanul Ferdaus[1], Manzur Murshed[2], Rodrigo N. Calheiros[3], Rajkumar Buyya[3]

[1] Faculty of Information Technology, Monash University, Churchill Vic 3842, Australia.
[2] School of Information Technology, Faculty of Science, Federation University Australia, Churchill Vic 3842, Australia.
[3] Department of Computing and Information Systems, The University of Melbourne, Australia.

Abstract. In this paper, we propose the AVVMC VM consolidation scheme that focuses on balanced resource utilization of servers across different computing resources (CPU, memory, and network I/O) with the goal of minimizing power consumption and resource wastage. Since the VM consolidation problem is strictly NP-hard and computationally infeasible for large data centers, we propose adaptation and integration of the Ant Colony Optimization (ACO) metaheuristic with balanced usage of computing resources based on vector algebra. Our simulation results show that AVVMC outperforms existing methods and achieves improvement in both energy consumption and resource wastage reduction.

1 Introduction

Cloud computing, a very recent paradigm shift in IT industry, is growing rapidly with the goal of providing virtually infinite amount of computing, storage, and communication resources where customers are provisioned these resources according to their demands as a pay-per-use business model [1]. To meet the rapid growth of customer demands for computing power, cloud providers such as Amazon and Google are deploying large number of planet-scale power-hungry data centers across the world, even comprising more than 1 million servers [2]. Reports show that energy is one of the critical TCO (Total Cost of Ownership) variables in managing a data center, and servers and data equipment account for 55% of energy used by data centers [3]. Large data centers also have enormous effects on the environment: higher energy consumption consequently drive in more carbon emission. Furthermore, inefficient use is one of the key factors for the extremely high energy consumption: in traditional data centers, on average servers operate only at 10-15% of their full capacity most of the time, leading to expenses on over-provisioning of resources [4]. Since cloud promises virtually unlimited resources through elastic provisioning and absolute reliability and availability, over-provisioning of resources in cloud data centers is a common phenomenon.

Virtualization technologies allow data centers to address such resource and energy inefficiency by placing multiple Virtual Machines (VM) in a single physical server through live VM migration techniques. Reduction of energy consumption is achieved by switching idle physical servers to lower power states (e.g., suspended) while still preserving application performance requirements.

In this paper, we propose AVVMC, a VM consolidation algorithm that focuses on balanced resource utilization of servers for different resource types. We present adaptation techniques of the popular *Ant Colony Optimization* (ACO) [5] metaheuristic with vector algebra-based multi-dimensional server resource utilization capturing method [6]. Through simulation-based evaluation, we show that AVVMC outperforms four other existing VM consolidation methods in different performance metrics.

2 Related Works

VM consolidation techniques have been very attractive to reduce energy costs and increase resource utilization in virtualized data centers. Consequently, a good amount of research works have been done in this area and depending on the modeling techniques used, different problem solving techniques are proposed. Most of the works that apply greedy heuristics primarily model VM consolidation as variants of the bin packing problem and propose extensions of simple greedy algorithms such as First Fit Decreasing (FFD) [7], Best Fit [6], Best Fit Decreasing [8], and so on [9, 10]. However, as VM consolidation is a NP-hard problem, greedy approaches are not guaranteed to generate near optimal solutions. Moreover, most of the approaches use mean estimators that fail to capture the multi-dimensional aspect of server resource utilization [6].

Using constraint programming (CP) model, Van et al. [11] proposed VM provisioning and placement techniques to achieve high VM packing efficiency in cloud data centers. Entropy [12] is a server consolidation manager proposed for clusters with the goal of minimizing the number of active servers and VM migration overhead. However, by the use of CP the proposed frameworks effectively restrict the domain of the total number of servers and VMs in data center, and thus limit the search space.

Recently, ACO metaheuristics have successfully been used to address 1-dimensional bin packing problem and VM consolidation. Levine et al. [13] first proposed an ACO-based solution for bin packing problem combined with a local search algorithm. Later, Brugger et al. [14] used a later version of the ACO metaheuristic that demonstrated superior performance over genetic algorithm for large problem instances. Feller et al. [15] used another version of ACO to address VM consolidation and has shown better results than FFD. However, the evaluation is shown by varying only the number of cores demanded by VMs while keeping other resource demands unchanged and as a result the evaluation is simplified to one-dimensional resource. Another recent work [16] proposed a multi-objective ACO algorithm to reduce resource wastage and power consumption in cloud data centers. This work considers two types of resources (i.e.

CPU and memory) and demonstrates performance improvement over genetic and other ACO-based algorithms.

3 Virtual Machine Consolidation

Most of the popular cloud providers offer different categories of VMs with specification for each type of resource. These VM instances differ in their individual resource capacity: some instances are larger than others, whereas some instances have relatively higher capacity of one type of resource compared to other resources. Moreover, cloud VM instances host various types of applications and active VMs exhibit dynamic resource demands in run-time that can be captured and used to perform workload prediction and estimation [17]. Because of the above properties of VM instances and dynamic workloads, complementary resource demands across difference resource dimensions are common in cloud data centers [6]. Furthermore, as clouds offer an on-demand pay-as-you-go business model, customers can demand for creation and termination of any number of VMs according to their requirements. As a result, VMs are created and terminated in the cloud data centers dynamically, which causes resource fragmentation in the servers, and thus leads to degradation in server resource utilization. VM consolidation is a tool to address the above issues in virtualized data centers that tries to pack the active VMs in the minimum number of physical servers considering multi-dimensional resource demands with the goal of energy saving and maximization of server resource utilization.

3.1 Modeling VM Consolidation as Multi-dimensional Vector Packing Problem

Multi-dimensional Vector Packing Problem (mDVPP) is a NP-hard combinatorial optimization problem where a number of items have to be packed into the minimum number of bins provided that bins capacities are not violated [18]. We model the physical machines (PMs) as bins and the VMs as items to pack into the bins. Let P denotes the set of n PMs and V denotes the set of m VMs in the data center. The set of d types of resources available in the PMs is represented by R. Each PM P_i ($P_i \in P$) has a d-dimensional *Resource Capacity Vector* (RCV) $C_i = \langle C_i^1, \ldots, C_i^k, \ldots, C_i^d \rangle$, where C_i^k denotes the total capacity of resource R_k of PM P_i. Similarly, each VM V_j ($V_j \in V$) is represented by its d-dimensional *Resource Demand Vector* (RDV) $D_j = \langle D_j^1, \ldots, D_j^k, \ldots, D_j^d \rangle$, where D_j^k denotes the demand of resource R_k of VM V_j. The *Resource Utilization Vector* (RUV) $U_i = \langle U_i^1, \ldots, U_i^k, \ldots, U_i^d \rangle$ of PM P_i is computed as the sum of the RDVs of the hosted VMs:

$$U_i^k = \sum D_j^k \text{ for } \forall x_{i,j} = 1 \tag{1}$$

where x is the *Placement Matrix* that models the VM-to-PM placements and is defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if } V_j \text{ is placed in } P_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We also introduce the *PM Allocation Vector* y, where each element y_i equals 1 if PM P_i is hosting at least 1 VM, or 0 otherwise:

$$y_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{m} x_{i,j} \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The goal of the AVVMC VM consolidation algorithm is to place the VMs in the available PMs in such a way that: 1) resource utilization of active PMs is maximized across all dimensions and 2) power consumption of active PMs is minimized. Since available models for server power consumption primarily focus on CPU utilization [19], any placement decision that results in lesser number of active PMs compared to others have higher resource utilization across all dimensions and lesser energy consumption. So, we formulate the objective function f as a single minimization function on y:

$$minf(y) = \sum_{i=0}^{n} y_i \tag{4}$$

Finally, the PM resource capacity constraint (i.e. for each resource type, demands D^k of hosted VMs not to exceed host PM's resource capacity C^k) is expressed as follows:

$$\sum_{j=1}^{m} D_j^k x_{i,j} \leq C_i^k, \forall i \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, d\} \tag{5}$$

And the following ensures that each VM is assigned to at most one PM:

$$\sum_{i=1}^{n} x_{i,j} \leq 1, \forall j \in \{1, \ldots, m\} \tag{6}$$

3.2 Modeling Multi-dimensional Resource Utilization based on Vector Algebra

When placing VMs in a PM, capturing the measure of overall resource utilization for multiple resource types is one of the most important factors for any server consolidation algorithm: saturation of only one resource type can lead to no further improvement in utilization while leaving other types of resource underutilized. In order to capture both balanced and overall resource utilization, we augment and integrate the vector algebra-based complementary resource utilization capturing technique [6] in our ACO-based solution. Our model considers CPU, memory, and network I/O as relevant server resources in the context of VM consolidation. We consider storage resource is provided on-demand through SAN/NAS-based storage backbone (e.g., Amazon EBS). PM's normalized resource capacity is expressed as a unit cube (*Resource Cube*), with the three dimensions representing three types of resources. RCV and RUV represent the

total capacity and current resource utilization of PM, respectively. To capture the degree of imbalance in current resource utilization of a PM, the *Resource Imbalance Vector* (RIV) is used which is computed as vector difference between RUV's projection on RCV and RUV itself. Given $RUV = C\hat{i} + M\hat{j} + I\hat{k}$ of a PM after placing a VM (C, M, and I are current utilization of CPU, memory, and network I/O), $RIV = (C - H)\,\hat{i} + (M - H)\,\hat{j} + (I - H)\,\hat{k}$, where $H = (C + M + I)/3$. When selecting VMs for placement in a PM, the VM that shortens the magnitude of RIV most is the VM that mostly balances the resource utilization of the PM across different dimensions. The magnitude of RIV is given by the following:

$$magRIV = \sqrt{(C - H)^2 + (M - H)^2 + (I - H)^2} \tag{7}$$

We use $magRIV$ to define the heuristic information for the proposed AVVMC algorithm along with the overall resource utilization of PM (Eq. 13).

3.3 Modeling Resource Utilization and Wastage

The overall resource utilization of PM p is modeled as the summation of the normalized resource utilization U_p^r of each individual resource $r \in R$ (Eq. 1): $Utilization_p = \sum_{r \in R} U_p^r$, where $R = \{CPU, MEM, IO\}$. Similarly, resource wastage is modeled as the summation of the remaining resources (normalized) of each individual resource:

$$Wastage_p = \sum_{r \in R} (1 - U_p^r) \tag{8}$$

3.4 Modeling Power Consumption

Power consumption of servers is dominated by their CPU and can be expressed as a linear expression of CPU utilization [19]. So, we model the energy drawn by a PM p as a linear function of its CPU utilization $U_p^{CPU} \in [0, 1]$:

$$E(p) = \begin{cases} E_{idle} + (E_{full} - E_{idle}) \times U_p^{CPU} & \text{if } U_p^{CPU} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where E_{full} and E_{idle} are the average energy drawn when a PM is fully utilized (i.e. 100% CPU busy) and idle, respectively. Due to the non-proportional power usage (i.e. high idle power) of physical servers, we consider turning off or suspending idle servers after the VM placement. Therefore, the estimate of the total energy consumed by a VM placement decision x is computed as follows:

$$E(x) = \sum_{p=1}^{n} E(p) \tag{10}$$

4 Proposed Solution

4.1 Adaptation of ACO Metaheuristic for VM Consolidation

ACO metaheuristics are computational methods that take inspiration from the foraging behavior of some ant species [5]. In ACO, a number of artificial ants build solutions to the considered optimization problem by choosing feasible solution components and exchanging information on the quality of these solutions via pheromone. In the proposed AVVMC algorithm, we adapt *Ant Colony System* (ACS) [20], a later version of ACO and consider each VM-to-PM assignment as a solution component. Pheromone levels are associated to all VM-to-PM assignments representing the desirability of assigning a VM to a PM (Eq. 11 and Eq. 18) and heuristic values are computed dynamically for each VM-to-PM assignment representing the favorability of assigning a VM to a PM in terms of both overall and balanced resource utilization of the PM (Eq. 13).

4.2 AVVMC Algorithm

The AVVMC algorithm pseudocode is shown in Algorithm 1. Pheromone levels are implemented using a $n \times m$ matrix τ. Each ant starts with an empty solution, a set of PMs, and a randomly shuffled set of VMs [line 6-12]. Inside the while loop, an ant is chosen at random and is allowed to choose a VM to assign next to its current PM among all the feasible VMs (Eq. 16) using a probabilistic decision rule (Eq. 15) [line 11-22]. If the current PM is fully utilized or there are no feasible VMs left to assign to the PM, a new empty PM is taken to fill in [line 14-16].

When all the ants have finished building their solutions, all the solutions in the current cycle are compared to the so far found *global-best-solution* (GBS) against their achieved objective function values f (Eq. 4). The solution with minimum value of f is chosen as the current GBS [line 23-28]. The pheromone reinforcement amount is computed based on (Eq. 19) and the pheromone levels of each VM-PM pair is updated to simulate the pheromone evaporation and deposition according to (Eq. 18) [line 29-34]. The algorithm reinforces the pheromone values only on the VM-PM pairs that belong to the GBS. Afterwards, the whole process of searching new solutions repeats. The algorithm terminates when no further improvement in the solution quality is observed for the last $nCycleTerm$ cycles [line 35]. Different parts of the algorithm are formally defined below.

Definition of Pheromone and Initial Pheromone Amount At the beginning of any ACO algorithm, ants start with a fixed amount of initial pheromone for each VM-PM solution component. Following the approach used in the original ACS algorithm [20], we use the measure of quality of the solution produced by a reference baseline algorithm (FFD heuristic based on L_1 norm mean estimator) to compute the initial amount of pheromone:

$$\tau_0 := PE_{FFDL1Norm} \tag{11}$$

Algorithm 1 The AVVMC Algorithm.

Input: Set of PMs P and their RCV C_i, set of VMs V and their RDV D_j, set of ants $antSet$. Set of parameters $\{nAnts, nCycleTerm, \beta, \omega, \delta, q_0\}$
Output: Global-best-solution GBS
Initialize parameters, set pheromone value for each VM-PM pair ($\tau_{v,p}$) to τ_0, $GBS := \emptyset, nCycle := 0$
repeat
  for all $ant \in antSet$ **do**
    $ant.solution := \emptyset; ant.pmList := P$
    $ant.p := 1; ant.vmList := V$
    Shuffle $ant.vmList$
  end for
  $antList := antSet; nCycle := nCycle + 1$
  while $antList \neq \emptyset$ **do**
    pick an ant at random from $antList$
    if $ant.vmList \neq \emptyset$ **then**
      if $FV_{ant}(ant.p) = \emptyset$ **then**
        $ant.p := ant.p + 1$
      end if
      Choose a VM v from $FV_{ant}(ant.p)$ accord. to Eq. 15
      $ant.solution.x_{p,v} := 1; ant.vmList.remove(v)$
    else
      $ant.solution.f := p; antList.remove(ant)$
    end if
  end while
  for all $ant \in antSet$ **do**
    if $ant.solution.f < GBS.f$ **then**
      $GBS := ant.solution$
      $nCycle := 0$
    end if
  end for
  Compute $\Delta\tau$
  for all $p \in P$ **do**
    for all $v \in V$ **do**
      $\tau_{v,p} := (1 - \delta) \times \tau_{v,p} + \delta \times \Delta\tau_{v,p}$
    end for
  end for
until $nCycle = nCycleTerm$

where $PE_{FFDL1Norm}$ is the *Packing Efficiency* of the solution produced by the FFD heuristic. The PE of any solution sol produced by an algorithm is given by:

$$PE_{sol} = \frac{nVM}{nActivePM} \tag{12}$$

Definition of Heuristic Information During the solution building process, the heuristic value $\eta_{v,p}$ represents a measure of benefit of selecting a solution component $v - p$. As the goal of AVVMC is to reduce the number of active PMs by packing VMs in a balanced way, we define the heuristic value favoring both balanced resource utilization in all dimensions and higher overall resource utilization:

$$\eta_{v,p} = \omega \times |log_{10} magRIV_p(v)| + (1 - \omega) \times Utilization_p(v) \tag{13}$$

where $magRIV_p(v)$ is the magnitude of RIV of PM p after assigning VM v to it (Eq. 7). Logarithm of $magRIV_p(v)$ is taken to give higher heuristic values to the v-p pairs that result in smaller magnitudes of RIV. $Utilization_p(v)$ is the

overall resource utilization of PM p if VM v is assigned to it:

$$Utilization_p(v) = \sum_{r \in R} (U_p^r + D_v^r) \tag{14}$$

And ω is a parameter that trades off the relative importance of balanced versus overall resource utilization as per our definition.

It can be shown that $magRIV$ is in the interval $[0.0, 0.82]$. Since logarithm of zero is undefined, we used the range $[0.001, 0.82]$ in the evaluation and thus $|log_{10} magRIV|$ results in the range $[0.086, 3.0]$ which is compatible to $Utilization_p$ in terms of metric that results in the interval $[0.0, 3.0]$.

Pseudo-random Proportional Rule When constructing a solution, an ant k selects a VM s to be assigned to PM p with the following *pseudo-random proportional rule* [20]:

$$s = \begin{cases} argmax_{v \in FV_k(p)} \{\tau_{v,p} \times [\eta_{v,p}]^\beta\} & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{15}$$

where q is a random number uniformly distributed in $[0, 1]$, q_0 is a parameter in interval $[0, 1]$, $\tau_{v,p}$ is the current pheromone value associated with the v-p VM-PM pair (Eq. 18), and β is a non-negative parameter that determines the relative importance of pheromone amount versus heuristic value in the decision rule. $FV_k(p)$ defines the list of feasible VMs for ant k to assign to PM p:

$$FV_k(p) = \left\{ v | \sum_{p=1}^{n} x_{p,v} = 0 \bigwedge U_p^r + D_v^r \leq C_p^r \text{ for } \forall_r \in R \right\} \tag{16}$$

When $q \leq q_0$, then the v-p pair resulting highest $\tau_{v,p} \times [\eta_{v,p}]^\beta$ value is chosen as the solution component (exploitation), otherwise a VM v is chosen with probability $P_k(v, p)$ using the following random-proportional rule (exploration):

$$P_k(v,p) = \begin{cases} \frac{\tau_{v,p} \times [\eta_{v,p}]^\beta}{\sum_{u \in FV_k(p)} \tau_{v,p} \times [\eta_{v,p}]^\beta} & \text{if } v \in FV_k(p) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Global Pheromone Update In order to favor the solution components of the GBS for subsequent iterations and to simulate pheromone evaporation, the global update rule is applied on the pheromone values of each $v-p$ pair according to the following equation:

$$\tau_{v,p} := (1 - \delta) \times \tau_{v,p} + \delta \times \Delta\tau_{v,p} \tag{18}$$

where δ is the global pheromone decay parameter ($0 < \delta < 1$) and $\Delta\tau_{v,p}$ is the pheromone reinforcement applied to each $v-p$ pairs and is computed as follows:

$$\Delta\tau_{v,p} = \begin{cases} PE_{GBS} & \text{if } v - p \in GBS \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

5 Performance Evaluation

Because of the lack of access to large scale testbeds or real cloud infrastructures and ease of reproducibility, we resorted to simulation-based evaluation to compare the performance of the proposed AVVMC to the following existing works in literature: 1) an adapted version of Max-Min Ant System (MMAS) metaheuristic for VM consolidation (MMVMC) [15], 2) a greedy algorithm (VectorGreedy) [6] for solving consolidation that uses vector algebra for mean estimation of multi-dimensional resources, 3) a modified version of the FFD algorithm (FFD-Volume) [7] that uses volume-based mean estimator, and 4) another modified FFD algorithm (FFD-L1Norm) based on L_1 norm mean estimator.

The simulated data center consists of a cluster of homogeneous PMs and VM resource demand for each resource type is expressed in percentage of total resource capacity of PM. We used reference-based VM resource demands: $Ref = z\%$ means each randomly generated VM resource demand D^r falls in the interval $[0, 2z]$ for $r \in \{CPU, MEM, IO\}$. Considering the fact that clouds deploy high-end servers and try to host as many VMs as possible in each active server to increase resource utilization, we conducted our simulation for the scenarios where expected average PE would be more than 4, otherwise there would not be much scope for consolidation and benefit of using specialized algorithms. Therefore, we used reference values of $Ref = 10\%$, 15%, 20%, and 25% with their corresponding expected average PE of 10, 6.7, 5, and 4. The simulation is conducted through 10 independent simulation runs and each run was repeated for 100 times and finally, the results are generated after taking their average.

The optimal values of the parameters used in AVVMC are measured through rigorous parameter sensitivity analysis in the preliminary phases of the experiment and are set as follows: $nAnts = 5, nCycleTerm = 5, \beta = 2, \delta = 0.5, q_0 = 0.8,$ and $\omega = 0.5$. Parameters for the other algorithms are taken as reported in the respective papers.

Table 1 summarizes performance of various algorithms for 1000 VMs in terms of 1) the number of active PMs, 2) achieved VM packing efficiency, and 3) power consumption according to the overall power consumption model (Eq. 10). For the purpose of simulation, we set E_{idle} and E_{full} to 162 watts and 215 watts, respectively as used by Gao et al. [16]. Table 1 shows that for all the four reference values, AVVMC outperforms other algorithms in all the above performance metrics. It also shows that AVVMC achieves PE near the expected average values. One interesting observation from column 6 of Table 1 is that AVVMC achieves comparatively better performance over MMVMC and VectorGreedy for larger reference values (i.e. larger VM sizes), whereas it achieves comparatively better performance over FFD-based algorithms for smaller reference values (i.e. smaller VM sizes). The reason is that metaheuristic-based solutions have higher flexibility to refine the solutions for smaller reference values (i.e. when higher number of VMs can be packed in a single PM) compared to larger reference values. On the other hand, FFD-based greedy solutions achieve higher overall resource utilization and need lesser number of active PMs for larger reference values (i.e. when VMs are larger).

Table 1. Simulation results across various performance metrics.

Ref	Algorithm	# Active PM	Achieved PE	Power Con. (Watt)	% Imp. (Power)
10%	AVVMC	100	10.00	21280.03	
	MMVMC	103	9.71	21759.55	2.20
	VectorGreedy	108	9.26	22582.51	5.77
	FFDL1Norm	117	8.55	23927.11	11.06
	FFDVolume	118	8.47	24165.25	11.94
15%	AVVMC	156	6.41	33114.59	
	MMVMC	163	6.13	34331.21	3.54
	VectorGreedy	167	5.99	34990.55	5.36
	FFDL1Norm	178	5.62	36824.39	10.07
	FFDVolume	177	5.65	36594.35	9.51
20%	AVVMC	215	4.65	45244.68	
	MMVMC	226	4.42	46945.68	3.62
	VectorGreedy	240	4.17	49225.02	8.09
	FFDL1Norm	242	4.13	49628.40	8.83
	FFDVolume	242	4.13	49677.00	8.92
25%	AVVMC	267	3.75	56325.08	
	MMVMC	286	3.50	59438.72	5.24
	VectorGreedy	310	3.23	63289.46	11.00
	FFDL1Norm	296	3.38	61008.50	7.68
	FFDVolume	296	3.38	61099.22	7.81

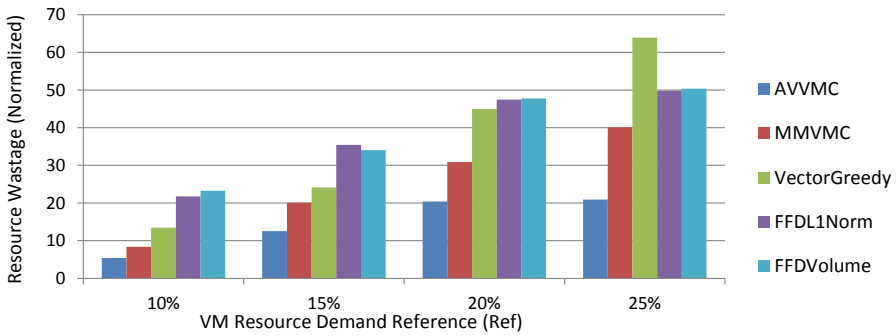

Fig. 1. Bar chart representation of total resource (normalized) wastage of AVVMC and other algorithms.

Fig. 1 shows a bar chart representation of the total resource (normalized in percentage) wastage of active PMs that host 1000 VMs according to (Eq. 8) for the VM placement solutions produced by the different consolidation algorithms. The figure shows that AVVMC significantly reduces the resource wastage compared to other algorithms. This is because AVVMC tries to improve the overall resource utilization with preference to consolidate VMs with complementary resource demands in each server and thus reduces resource wastage across different resource dimensions.

In order to assess AVVMC for time complexity, simulation is conducted for larger number of VMs and the solution computation time is plotted (Fig. 2). The algorithm is written in Java language and ran on a Dell Workstation having Intel Core i5-2400 3.10 GHz CPU (4 cores), and 4 GB of RAM. It is observed that computation time increases non-linearly with the number of VMs with small gradient and for 2000 VMs, AVVMC takes around 25 seconds on average.

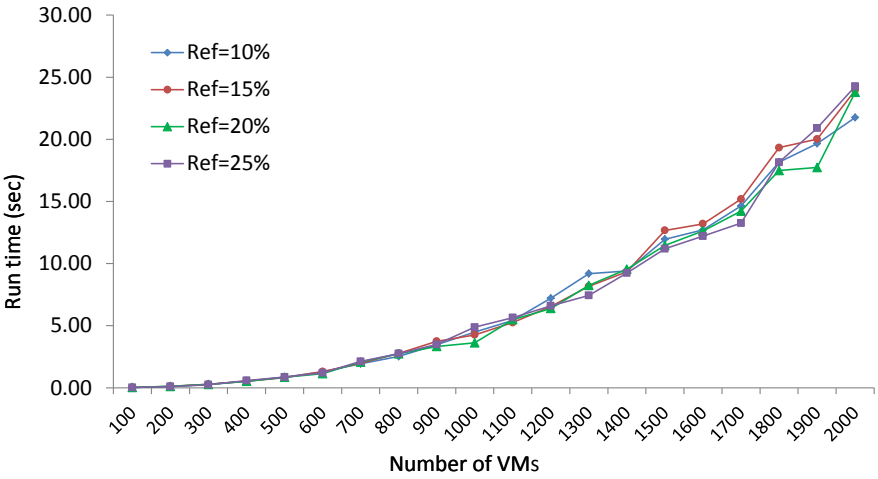

Fig. 2. Solution computation time of AVVMC for large problem instances.

6 Conclusions and Future Work

In this paper, we presented several aspects of server resource utilization and consolidation, and proposed an ACO metaheuristic-based server consolidation mechanism to address both power consumption and resource wastage minimization in large virtualized data centers. We presented performance evaluation by comparing the proposed technique with some of the recent techniques proposed in the literature. We also showed evaluation of time complexity of solution computation and argued about the feasibility and effectiveness of the algorithm for cloud data centers.

As future work, we plan to incorporate mechanisms for efficient network resource utilization in cloud infrastructures during VM placement and consolidation decisions. We also expect to consider current VM assignments and reconfiguration (including VM live migrations) overheads during VM placement decision making phase. In this way, an overall VM placement framework will be designed and implemented that will be aware of both energy consumption and compute-network resource utilization.

References

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems **25**(6) (2009) 599–616
2. Miller, R.: Ballmer: Microsoft has 1 million servers. http://www.datacenterknowledge.com/archives/2013/07/15/ballmer-microsoft-has-1-million-servers/ (July 2013)
3. Perspectives, I.: Using a Total Cost of Ownership (TCO) model for your data center. http://www.datacenterknowledge.com/archives/2013/10/01/using-a-total-cost-of-ownership-tco-model-for-your-data-center/ (October 2013)

4. Barroso, L., Holzle, U.: The case for energy-proportional computing. Computer **40**(12) (2007) 33–37
5. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. Computational Intelligence Magazine, IEEE **1**(4) (nov. 2006) 28–39
6. Mishra, M., Sahoo, A.: On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on, IEEE (2011) 275–282
7. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Sandpiper: Black-box and gray-box resource management for virtual machines. Computer Networks **53**(17) (2009) 2923–2938
8. Beloglazov, A., Buyya, R.: Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In: Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, ACM (2010)
9. Li, X., Qian, Z., Chi, R., Zhang, B., Lu, S.: Balancing resource utilization for continuous virtual machine requests in clouds. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, IEEE (2012) 266–273
10. Li, X., Qian, Z., Lu, S., Wu, J.: Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Mathematical and Computer Modelling **58**(5) (2013) 1222–1235
11. Van, H.N., Tran, F., Menaud, J.M.: Performance and power management for cloud infrastructures. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. (july 2010) 329–336
12. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. VEE '09, New York, NY, USA, ACM (2009) 41–50
13. Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society **55**(7) (2004) 705–716
14. Brugger, B., Doerner, K., Hartl, R., Reimann, M.: Antpacking-an ant colony optimization approach for the one-dimensional bin packing problem. Evolutionary Computation in Combinatorial Optimization (2004) 41–50
15. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, IEEE Computer Society (2011) 26–33
16. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences (2013)
17. Wood, T., Cherkasova, L., Ozonat, K., Shenoy, P.: Predicting application resource requirements in virtual environments. HP Laboratories, Technical Report HPL-2008-122 (2008)
18. Caprara, A., Toth, P.: Lower bounds and algorithms for the 2-dimensional vector packing problem. Discrete Applied Mathematics **111**(3) (2001) 231–262
19. Fan, X., Weber, W.D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. ACM SIGARCH Computer Architecture News **35**(2) (2007) 13–23
20. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1) (apr 1997) 53–66