# Heterogeneity Aware Dominant Resource Assistant Heuristics for Virtual Machine Consolidation

Yan Zhang and Nirwan Ansari
Advanced Networking Laboratory, Department of Electrical and Computer Engineering,
New Jersey Institute of Technology, Newark, NJ 07012, United States
{yz45, nirwan.ansari}@njit.edu

***Abstract*—Power consumption is a critically important issue for data centers. Virtual machine (VM) consolidation is fundamentally employed to improve resource utilization and power optimization in modern data centers. In general, VM consolidation is formulated as a vector bin packing problem, which is a well-known NP-hard problem. Hence, heuristic algorithms, such as single dimensional heuristics (e.g., first fit decreasing (FFD)) and dimension-aware heuristics (e.g., DotProduct), are usually deployed in practice for VM consolidation. However, all of these previous heuristic algorithms did not sufficiently explore the heterogeneity of the VMs' resource requirements. In this paper, we propose several heterogeneity aware dominant resource assistant heuristic algorithms for VM consolidation. The performance evaluations validate the effects of the proposed heterogeneity aware heuristics on VM consolidation. The proposed heuristics can achieve quite similar consolidation performance as dimension-aware heuristics with almost the same computational cost as those of the single dimensional heuristics.**

***Index Terms*—Virtual machine consolidation, data center, cloud computing, first fit decreasing (FFD), dominant resource.**

## I. Introduction

The rapid escalating power consumption of data centers has become a critically important issue in data centers [1]. One of the main power guzzlers in data centers is the servers, which contribute to 30-50% power consumption of the data centers [2–4]. Virtual machine (VM) consolidation is an important technique for efficient usage of server resources to reduce the total number of servers, thus potentially resulting in reducing the power consumption of data centers. VM consolidation is often modeled as vector bin packing problems, which are well-known to be NP-hard [5]. Therefore, heuristic algorithms are usually used to solve this type of problems [6–8].

First fit decreasing (FFD) and best fit decreasing (BFD) are the most popular heuristic algorithms to solve the VM consolidation problem. Verma *et al.* [6] investigated a power and migration cost aware application placement framework pMapper to minimize power and migration costs while meeting the performance guarantees in heterogeneous server clusters. They also proposed an improved FFD heuristic algorithm, namely, min power parity (mPP), to map VMs to physical machines (PMs). The basic idea of mPP is to allocate VMs to the most energy-efficient PM, thus resulting in the least power increase per unit increase in resource utilization. Beloglazov and Buyya [7] applied BFD for VM placement in their proposed energy efficient resource management system for virtualized cloud data centers. Similar to mPP, a modified best fit decreasing (MBFD) heuristic algorithm [8] was designed to consolidate VMs onto the most energy-efficient PMs. The main difference between mPP and MBFD is that mPP sorts PMs in the descending order of power efficiency, while MBFD does not.

Besides FFD and BFD, a minimum bin slack based heuristic, namely, incremental power aware consolidation (IPAC), was proposed in [9]. For a given server, IPAC allocates a subset of unallocated VMs to maximize the server's CPU utilization constrained by the server's resource capacity. As shown in [9], IPAC provides a better solution in terms of power consumption than FFD does, but the computation complexity to solve the minimum slack problem may limit its application in practice. Computational intelligence [10, 11], such as genetic algorithms [12–14] and simulated annealing [15], has also been deployed to improve the VM consolidation performance. However, these algorithms are computational expensive, and generally take more time to reach the optimal result. Hence, we focus on developing fast and efficient heuristic algorithms for VM consolidation.

In this paper, we propose several heuristic algorithms which explore the heterogeneity of the VMs' requirements for different resources and utilize the dominant resource (i.e., the resource corresponding to the largest element in the vector of the normalized VM's resource demand over the average resource capacity vector of all PMs) to assist VM consolidation. The heterogeneity of the VMs' requirements for different resources captures the differences among VMs' demands, and the dominant resource of a VM or a PM captures the primary request of a VM or the primary resource of a PM. These two characteristics have not been considered in the previous heuristic algorithms for VM consolidation. The remainder of this paper is organized as follows. In the next section, we formulate the VM placement problem. Some popular heuristics for VM placement are described in Section III. The proposed several dominant resource aware heuristics are presented in Section IV. The performance of the proposed heuristic algorithms for VM placement is evaluated in Section V. Finally, Section VI concludes the paper.

## II. Virtual Machine Consolidation Problem

Most works published in the research literature formulate the VM consolidation as an optimization problem to minimize the number of active servers with constraints imposed by server capacities, service level agreements, etc. Assume that


978-1-4799-1353-4/13/$31.00 ©2013 IEEE

$K$ VMs $\{V_1, \cdots, V_K\}$ are requested by clients and assigned to a data center consisting of $M$ distinct host physical machines $\{M_i\}$ ($i \in [1, M]$). Each PM $M_i$ can be characterized by a $d$-dimensional resource capacity vector $C_i = [C_{i1}, \cdots, C_{id}]$ ($i \in [1, M]$). Each dimension corresponds to a different resource such CPU, memory, network bandwidth, or disk bandwidth. Similarly, each VM is characterized by a $d$-dimensional resource request vector $R_k = [r_{k1}, \cdots, r_{kd}]$, where $r_{kj}$ ($j \in [1, d]$) is the VM $V_k$'s resource demand for dimension $j$. The VM consolidation algorithm determines how to allocate these $K$ VMs on $M$ PMs, denoted by $x_{ij} = \{0, 1\}$ ($i \in [1, K], j \in [1, M]$) which is the binary variable indicating whether the VM $V_i$ is assigned to PM $M_j$. If the VM $V_i$ is assigned to PM $M_j$, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

Based on the above definitions, the goal of the VM consolidation algorithms is to place the VMs on as few as possible active PMs, while at the same time ensuring that the total resource demands of VMs allocated on any active PM does not exceed its resource capacity across any dimension. If PM $M_j$ has been assigned one or more VMs, this PM is said to be active, $y_j = 1$; otherwise, it stays at the idle state, $y_j = 0$. Thus, the objective function of the VM consolidation algorithm is to minimize the total number of active PMs as:

$$\min \sum_{j=1}^{M} y_j \tag{1}$$

subject to

1) Resource capacity constraints: for each resource of one PM $M_j$, the total quantity utilized by VMs allocated to this PM cannot exceed its corresponding resource capacity weighed by its maximum resource utilization factor $\gamma_{jk}$ ($k \in [1, d]$).

$$\sum_{i=1}^{K} r_{ik} \cdot x_{ij} \leq \gamma_{jk} \cdot C_{jk} \quad (\forall j \in [1, M], \forall k \in [1, d]) \tag{2}$$

2) Placement guarantee constraints: each VM can only be assigned to one PM.

$$\sum_{j=1}^{M} x_{ij} = 1 \quad (\forall i \in [1, K]) \tag{3}$$

3) Correlation between the binary status of PMs $y_j$ with VMs allocation $x_{ij}$: for each PM $M_j$, its binary status $y_j$ should not be smaller than the binary allocation status of VM $V_i$ allocated to PM $M_j$ $x_{ij}$, since any one of the VM $V_i$ assigned to PM $M_j$ will result in $y_j = 1$.

$$y_j \geq x_{ij} \quad (\forall i \in [1, K], \forall j \in [1, M]) \tag{4}$$

## III. Heuristics for Virtual Machine Consolidation

In this section, some popular heuristic algorithms for virtual machine consolidation will be reviewed and discussed. Based on the scale used for VM assignment, heuristic algorithms for VM consolidation can be classified into two categories: single dimensional heuristics and dimension aware heuristics.

### A. Single Dimensional Heuristics

The basic idea behind the single dimensional heuristics is mapping the vector of PM's resource capacities and the VM's resource demands into a single scalar, which will be used to perform VM consolidation while ignoring the relationships across dimensions.

*1) First Fit Decreasing (FFD):* FFD heuristic sorts the VMs and PMs in the non-increasing order of their "size". It could be the normalized resource demand or capacity in any one of the dimensions (e.g., CPU, memory), or the weighted sum of normalized resource demand or normalized resource capacity across all dimensions, or the product of normalized demand or normalized capacity across all dimensions [16]. The FFD heuristic algorithms based on these different sorting rules are denoted as *FFDSDim*, *FFDSum*, and *FFDProd*, respectively, throughout this paper. Hence, the VM "size" can be defined as follows:

$$Size(V_i) = \begin{cases} r_i & (\textit{FFDSDim}) \\ \sum_{j=1}^{d} w_j r_{ij} & (\textit{FFDSum}) \\ \prod_{j=1}^{d} r_{ij} & (\textit{FFDProd}) \end{cases} \tag{5}$$

where $\{w_j\}$ ($j \in [1, d]$) denotes the weight coefficients for different resources. Similarly, the PM's "size" can be defined by replacing the vector of the VM's resource requests with the vector of the PM's resource capacity.

Starting from the largest VM in size, FFD iteratively assign VMs to PMs. For each VM, FFD also starts from the largest PM in size to iteratively check the PM's residual capacity, which is defined as the PM's resource capacity subtracting the total resource demands of VMs assigned to this PM. The VM will be packed to the first PM which has enough residual capacity, implying that the residual capacity of this PM is larger than the VM's resource demand across any dimension. If none of active PMs can host the VM, the largest idle PM will be activated.

*2) Best Fit Decreasing (BFD):* Similar to FFD, the BFD heuristic also sorts the VMs in size, and iteratively assigns each VM to active PMs starting from the largest VM. The main difference between BFD and FFD is that BFD allocates the VM to PMs based on some rules, such as Least Full First (LFF) and Most Full First (MFF). With LFF, the VM will be assigned to the active PM with the least resource utilization. On the other hand, with MFF, the VM will be assigned to the active PM with the highest resource utilization.

### B. Dimension aware Heuristics

The main drawback of single dimensional heuristics is that the vectors of the PM's resource capacities and the VM's resource requests are mapped into a single scalar, and thus the PMs' complimentary requirements for different resources are not considered during the VM assignment procedure. Contrary to the single dimensional heuristics, the dimension aware heuristics attempt to take advantages of the PMs' complimentary resource requirements across all resource dimensions.

*1) DotProduct:* Singh *et al.* [17] proposed a load balancing scheme, called VectorDot, for handling the multi-dimensional resource constraints in data centers. Basically, VectorDot utilizes the dot product $\sum_i^d w_i r_i h_i(t)$ between the vector of a PM's residual capacities at time $t$, $H(t) = \{h_1(t), \cdots, h_d(t)\}$, and the vector of the VM's resource requirement elements $R_i = \{r_{i1}, \cdots, r_{id}\}$ across all dimensions to choose the target PM for VM placement, where $w_i$ is the weight parameters

calculated in the same manner as described in the FFDSum heuristic. This idea can also be used for VM consolidation, which is denoted as the DotProduct heuristic algorithm. Given a PM, DotProduct selects and allocates the VM that maximizes the dot product $\sum_i^d w_i r_i h_i(t)$ while not violating the PM's capacity constraints. The main idea of DotProduct is to select and assign a VM on a given PM for which the resource requirement vector of the VM is complementary to the resource utilization vector of the PM.

*2) Norm-based Greedy ($\mathcal{L}(2)$):* For each newly activated PM, the norm-based greedy heuristic iteratively selects one VM $V_i$ among all unassigned VMs that does not violate the PM's capacity constraint and minimizes the weighted norm distance $\sum_i w_i \|R_i - h_i(t)\|$ between the VM resource request vector $R_i$ and the residual capacity vector of the current PM $H(t) = [h_1(t), \cdots, h_d(t)]$ as well, until no VM can be packed into the current PM.

### C. Comparison

The comparison study among these heuristic algorithms has been performed in [18]. According to [18], *FFDSum* dominates other single dimensional heuristics and dimension aware heuristics can achieve up to 10% improvement over *FFDSum* with realistic workloads, and more consolidation gain can be obtained when VMs are mixed with different dominant resources, such as mix of CPU-intensive VMs and memory-intensive VMs. The heuristic comparison study [18] also shows that if all VMs are constrained by a single resource, the single dimensional heuristics are efficient and can provide almost the same results as dimension-aware heuristics.

## IV. Heterogeneity aware Dominant Resource assistant Heuristics

All of the heuristic algorithms discussed in the last section do not consider the heterogeneity of VMs' requirements for different resources. Inspired by the concept of "dominant resource" introduced in [19], which is the resource corresponding to the maximum among all requested resources of a user, we will introduce several heterogeneity aware dominant resource assistant heuristic algorithms in this section.

### A. Dominant Residual Resource aware FFD (DRR-FFD)

Inspired by the efficiency and VM consolidation performance of FFD heuristics, a new FFD derivative heuristic by considering the dominant residual resource, denoted as Dominant Residual Resource aware FFD (DRR-FFD), is proposed in this paper. The basic idea behind DRR-FFD is that it always tries to balance the resource utilizations across all dimensions by matching up the PM's residual resource capacity with the next VM to be allocated. For example, if more CPU resource than the memory resource is left, a CPU-intensive unassigned VM will be allocated to consume more CPU resource than memory resource. The DRR-FFD heuristic is summarized in Algorithm 1.

**Algorithm 1** DRR-FFD Heuristic

1: Cluster VMs by their dominant resources $\{G_1, \cdots, G_d\}$.
2: In each cluster, sort VMs in non-increasing size.
3: Sort PMs in the non-increasing order of the power efficiency.
4: Activate the most power efficient PM and assign the largest VM to it.
5: **while not** All VMs have been allocated **do**
6: Calculate the total residual capacity $\{\sum_j y_j h_{j1}, \cdots, \sum_j y_j h_{jd}\}$ and identify the dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$.
7: Allocate the largest unassigned VM in the non-increasing order of the identified dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$ with FFD heuristic.
8: **if** The selected VM in the last step is not assigned to any PM **then**
9: Activate the most power efficient idle PM and assign the selected VM to it.
10: **end if**
11: **end while**

Unlike FFD, DRR-FFD clusters the VMs based on their dominant resources, which are the resources corresponding to the largest requirements in the VMs' normalized $d$-dimensional resource request vectors (line 1 in Algorithm 1). In each cluster, the VMs are sorted in non-increasing order of the predefined VM "size" (line 2 in Algorithm 1), which can be defined based on different criteria as shown in Eq. (5). DRR-FFD also sorts PMs in non-increasing order of their power efficiency to ensure that VMs are always assigned to the subset of the most power efficient servers to reduce power consumption (line 3 in Algorithm 1). Similar to FFD, DRR-FFD starts from the largest VM (line 4 in Algorithm 1) and iteratively assigns one VM to the first PM which has enough residual resource capacity to host it until all the VMs have been allocated (lines 5-8 in Algorithm 1).

The major difference between FFD and DRR-FFD is that FFD sorts VMs in a single queue according to their sizes while DRR-FFD organizes VMs into multiple groups $\{G_1, \cdots, G_d\}$ based on the dominant resources. It is simple to determine the next VM to be assigned in FFD since all VMs are organized in a single queue, but DRR-FFD needs to provide a mechanism to determine the next unassigned VM to be allocated. DRR-FFD calculates the normalized residual resource capacities of all active PMs over their total resource capacities and identifies the dominant residual resource vector $D^R = \{D_1^R, \cdots, D_d^R\}$, which organizes the resources corresponding to the non-increasing order of their normalized residual resource capacities. For example, the normalized residual resource capacity of all active PMs is [0.2, 0.6] in CPU and memory, respectively. Hence, the dominant residual resource vector is $D^R$=[memory, CPU]. Then, DRR-FFD selects the largest unassigned VM in the memory cluster which has the largest residual resource capacity. If there is no unassigned VM in this cluster, DRR-

FFD will search for the cluster with the next largest residual resource capacity, which is CPU cluster in this example. DRR-FFD might run this step on several clusters until it gets a VM to be allocated. Also, as in the last example, DRR-FFD tries the cluster which contains all the memory-intensive VMs first. If this cluster is empty, it tries the cluster which contains all the CPU-intensive VMs.

### B. Individual DRR-FFD (iDRR-FFD)

One problem with DRR-FFD is that the sum of the residual capacities across all active PMs might mislead the actual dominant residual resource. Consider the case with five active PMs. Four of them have the normalized residual resource capacity [0.15 CPU, 0 memory], and the residual resource capacity of the fifth PM is [0.3 CPU, 0.5 memory]. Thus, the total residual resource capacity of these five PMs is [0.9 CPU, 0.5 memory], in which case DRR-FFD will select a CPU-intensive VM to be allocated. However, among these normalized CPU residual capacity, $67\%$ cannot be assigned to any VM because there is no enough memory to satisfy the resource capacity constraint. Also note that the utilization of the CPU resource is already higher than that of memory on the fifth PM. The allocation of another CPU-intensive VM to the fifth PM will even worsen the imbalance of resource utilizations. One solution to this problem is to use the residual capacity of each individual PM to identify the dominant residual resource vector. We refer to this heuristic as individual DRR-FFD (iDRR-FFD).

### C. Dominant Residual Resource based Bin Fill (DRR-BinFill)

Dominant Residual Resource based Bin Fill (DRR-BinFill) heuristic is quite similar to DRR-FFD, except that it focuses on a given server. The DRR-BinFill heuristic is summarized in Algorithm 2. Firstly, DRR-BinFill clusters VMs according to their dominant resources and sorts them in the non-increasing size (lines 1 and 2 in Algorithm 2), and also sorts PMs in the non-increasing order of their energy efficiency (line 3 in Algorithm 2), which are the same as Steps 1-3 of DRR-FFD, and therefore, the idle PMs can be activated in non-increasing order of their energy efficiency, thus ensuring that VMs are allocated to the subset of the most energy-efficient PMs to reduce power consumption (line 5 in Algorithm 2). The newly activated server is not necessary to be empty, meaning that no VM has been assigned to it. If the given server is empty, the largest unassigned VM will be allocated to it (line 6 in Algorithm 2); otherwise, this step will be skipped. Then, DRR-BinFill will iteratively allocate one VM among all the unassigned VMs to the given server to compensate for the server's residual resource capacity. We call this procedure as the bin filling process. In each iteration, DRR-BinFill calculates the given server's residual capacity and identifies the dominant residual capacity vector (line 9 in Algorithm 2), and searches for the VM clusters in the order of the server's dominant residual resource vector (line 13 in Algorithm 2), implying that the larger the given server's residual resource capacity, the earlier the corresponding VM cluster will be searched. The VM searching procedure will be stopped when one VM has been found without violating the given server's capacity constraints. If all clusters have been searched and such a VM cannot be found (lines 14-16 in Algorithm 2), or some of resources in the server have reached their capacity constraints (10-12), the bin filling process will be stopped.

**Algorithm 2** Dominant Residual Resource based Bin Fill

1: Cluster VMs by their dominant resources $\{G_1, \cdots, G_d\}$.
2: In each cluster, sort VMs in non-increasing size.
3: Sort PMs in the non-increasing order of the power efficiency.
4: **while not** All VMs has been allocated **do**
5: Activate the most energy efficient idle server $M_i$.
6: Allocate the largest unassigned VM to $M_i$.
7: **Comment:** The following is the Bin Filling Process.
8: **while not** Filling process finished **do**
9: Calculate the given server's residual capacity $h_j$ ($j \in [1, d]$) and identify the dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$.
10: **if** The residual capacity of one resource is smaller than $\epsilon$ **then**
11: break; //bin filling process finishes.
12: **end if**
13: Search for the clusters corresponding to the order of dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$ to get a VM to be allocated to the PM $M_i$ without violating the resource capacity constraints.
14: **if** Such a VM cannot be found **then**
15: break; //bin filling process finishes.
16: **end if**
17: **end while**
18: **end while**

## V. Performance Evaluation

We evaluate the performance of the proposed heterogeneity-aware heuristics under different synthetic workloads, and compare the results with other single dimensional heuristics and dimension aware heuristics.

### A. Synthetic Workload

Currently, we cannot obtain commercial VM traces due to privacy and security concerns. We generate several synthetic workloads to emulate VM demands and evaluate the proposed heuristics on these synthetic workloads. Most of these synthetic workloads are chosen from the ten classes used by Caprara and Toth [20] to benchmark the effectiveness of two-dimensional bin-packing heuristics. In the first VM class denoted by $Random[h, \alpha, \beta]$, the VM resource request in each dimension is drawn randomly and independently from the range $[\alpha, \beta]$, and the resource capacity of a PM in each dimension is $h$. In this class $Random[h, \alpha, \beta]$, the dimensions are independently generated and the parameters are set to $h = 150$ and $[\alpha, \beta] = [20, 100]$ as in the experiments

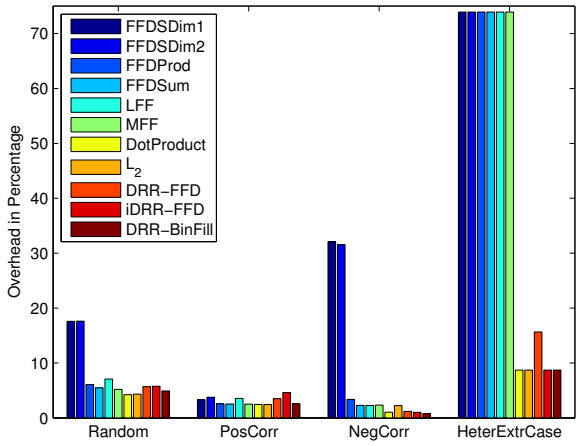


Fig. 1: VM consolidation performance of different heuristics.

performed in [18]. The dimensions in the other two classes, namely, positive correlation $PosCorr[h, \alpha, \beta]$ and negative correction $NegCorr[h, \alpha, \beta]$, are correlated. In both classes, PMs have $h = 150$ capacity in each resource dimension, and for each VM, the demand in the first dimension $r_1$ is sampled uniformly in the range [20, 100], while the demand in the second dimension $r_2$ is sampled uniformly in the range $[r_1 - 10, r_1 + 10]$ for the positive correlation class $PosCorr$ and in the range $[110-r_1, 130-r_1]$ for the negative correlation class $NegCorr$. The fourth class $HeterExtrCase$ is generated artificially to emulate the worst case to VMs consolidation in terms of the heterogeneity of the VMs'requirements for different resources. In this class, the PM capacity is set to $h = 100$ in each resource dimension, and VMs have two types of resource requests. One is the first resource dimension intensive VM request [20,1] and the other is the second resource dimension intensive VM request $[3, 20]$. Each type of VM request has 1000 VMs.

### B. VM Heterogeneity

We also investigate the effects of the heterogeneity of VMs' requirements for different resources to the consolidation performance of different heuristics. The ratio of the VM's demands for two different resources $\rho_i^{jk} = r_{ij}/r_{ik}$ $(j \neq k; j, k \in [1, d])$ can reflect the difference of the VM's demands for these two resources. Under the two-dimensional resource requirements, the heterogeneity of VMs' resource demands can be captured with the variance of the ratio of VM's demands for these two resources $\rho_i = r_{i1}/r_{i2}$. We limit the VMs' resource demands in two dimensions in this investigation. Assume the VM resource request in each dimension is drawn randomly and independently from the range $[1, 20]$. Hence, the ratio of the VM's demands for the two resources is in the range $[0.05, 20]$. Further assume the average of the resource demand ratio of all VMs is $\mu \in [0.05, 20]$. The ratio of VM's demands for the two resources is assumed to be a random variable with the normal distribution $\mathcal{N}(\mu, \sigma)$. The demand of the VM's first resource $r_{i1}$ is chosen randomly and uniformly in the range $[1, 20]$, and the demand of the VM's second resource is obtained with $r_{i2} = r_{i1}/\rho_i$. If $r_{i2}$ is in the range $[1, 20]$, the VM's resource demands $R_i = \{r_{i1}, r_{i2}\}$ will be kept; otherwise, the demand of the VM's first resource $r_{i1}$ will be reselected uniformly in the range $[1, 20]$, and the demand for the VM's second resource $r_{i2}$ will be recalculated accordingly.

### C. Results

The comparison study among different heuristic algorithms for VM consolidation is performed first with the four classes of synthetic workloads described above and the VM consolidation results are shown in Fig. 1 in terms of the percentage overhead of the number of active PMs calculated with different heuristic algorithms with respect to the lower bound $B_L$ of the active PMs under different workload scenarios. The lower bound of the number of PMs is estimated with the maximum number of PMs required to satisfy the total resource requests along any single resource dimension $\lceil \sum_{k=1}^{K} r_{ki}/C_i \rceil$ $(i \in [1, d])$ as follows:

$$B_L = max\{\lceil \textstyle\sum_{k=1}^{K} r_{k1}/C_1 \rceil, \cdots, \lceil \sum_{k=1}^{K} r_{kd}/C_d \rceil\} \tag{6}$$

From the results shown in Fig. 1, we can see that the consolidation performances of FFDProd, FFDSum, LFF, and MFF are very close to each other. All of these four single dimensional heuristics perform better than another single dimensional heuristic algorithm FFDSDim, the consolidation results of which based on the first resource dimension and the second resource dimension to sort the VMs is represented as "FFDSDim1" and "FFDSDim2", respectively. FFDSum performs a little bit better than FFDProd does across all of the four classes of synthetic workloads, implying that the weighted sum of the normalized resource demands is a better metric in sorting VMs in "size" than the product of the normalized

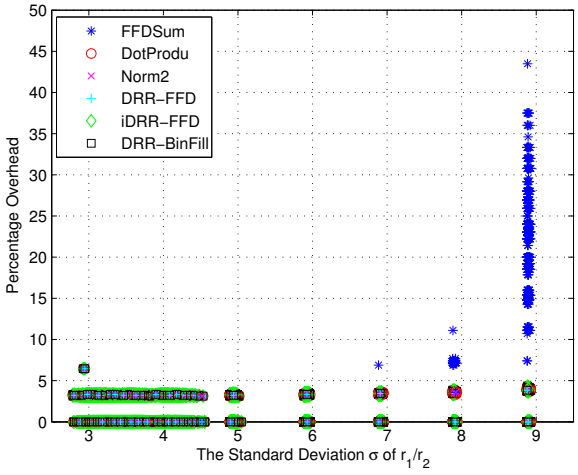


Fig. 2: VM consolidation performance over different resource requirement heterogeneity.

resource requirement elements. Therefore, the weighted sum of the normalized resource requirement elements will be used as the metric to sort VMs in other heuristics by default in this paper, unless otherwise stated. The consolidation performance of LFF is worse than that of MFF, because LFF also takes care of the load balance among the active PMs except VM consolidation. The dominant aware heuristics, DotProduct and Norm-based greedy $\mathcal{L}_2$, can achieve better consolidation performance than those of single dimensional heuristics, especially in the fourth synthetic workload scenario $HeterExtrCase$. Note that the variance of the ratios of the VM's requirements for two different resources is much larger than that in other three scenarios. The performances of DotProduct and Norm-based greedy $\mathcal{L}_2$ are nearly the same to each other. It is worth observing that the proposed heterogeneity aware dominant resource assistant heuristics, namely DRR-FFD, iDRR-FFD and DRR-BinFill, perform quite well in all of these four scenarios.

The performance of different heuristics under different VM heterogeneity in terms of the standard deviation of the resource ratios $\rho_i = r_{i1}/r_{i2}$ is shown in Fig. 2. This figure shows clearly that FFDSum performs poorly with large resource requirement ratios. Both the dimension aware heuristics, DotProduct and Norm-based greedy $\mathcal{L}_2$, and the heterogeneity aware dominant resource assistant heuristics, DRR-FFD, iDRR-FFD, and DRR-BinFill, perform quite well with large resource requirement ratios. As compared to the dimension aware heuristics, the proposed heterogeneity aware heuristics save the computations in calculating the dot product between the residual capacity vector of a PM and the VM resource request vector, or the weighted norm distance between these two vectors.

## VI. Conclusion

In this paper, we have proposed several heterogeneity-aware dominant resource assistant heuristic algorithms for virtual machine consolidation, which explore the heterogeneity of the virtual machine's requirements for different resources and utilize the dominant resources to assist VM consolidation. We have evaluated the performance of the proposed heuristic algorithms with different classes of synthetic workloads, and also investigated their performance under different VM requirement heterogeneity conditions. The simulation results have shown that the proposed heterogeneity-aware dominant resource assistant heuristics achieve better consolidation performance than single dimensional heuristics, i.e., FFDProd and FFDSum, with a little extra effort in clustering VMs according to their dominant resources and identifying the dominant residual resources. The results also show that the proposed heterogeneity-aware heuristics can achieve quite similar consolidation performance as dimension-aware heuristics with reduced computation.

## References


[1] U.S. Environmental Protection Agency, "Environmental Protection Agency, Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431," *ENERGY STAR Program*, Aug. 2007.

[2] G. Corp, "Reducing data center power and energy consumption: Saving money and "going green"," Jan. 2010. [Online]. Available: http://poweritdown.org/resources/wp_greeningdatacenter.pdf

[3] T. F. W. Steven Pelley, David Meisner and J. W. VanGilder, "Understanding and abstracting total data center power," in *Proc. of the 36th International Symposium on Computer Architecture (ISCA) Workshop on Energy Efficient Design (WEED)*, Austin, Texas, USA, June 20-24 2009.

[4] Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 39–64, First Quarter 2013.

[5] D. S. Hochbaum, *Approximation algorithms for NP-hard problems*. Boston, MA, USA: PWS Publishing Co., 1997.

[6] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proc. of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware'08)*, Leuven, Belgium, 2008, pp. 243–264.

[7] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *Proc. of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10)*, Melbourne, Australia, May 17-20, 2010, pp. 826–831.

[8] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, May 2012.

[9] Y. Wang and X. Wang, "Power Optimization with Performance Assurance for Multi-tier Applications in Virtualized Data Centers," in *Proc. of the 39th International Conference on Parallel Processing Workshops (ICPPW'10)*, San Diego, CA, USA, Sept. 13-16, 2010, pp. 512–519.

[10] N. Ansari and E. Hou, *Computational Intelligence for Optimization*. Springer, 1997.

[11] E. S. H. Hou, N. Ansari, and H. Ren, "A genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, Feb. 1994.

[12] J. Xu and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," in *Proc. of 2010 IEEE/ACM Internaltional Conference on Green Computing and Communications (GreenCom) & 2010 IEEE/ACM Internaltional Conference on Cyber, Physical and Social Computing (CPSCom)*, Hangzhou, China, Dec. 18-20, 2010, pp. 179–188.

[13] H. Hlavacs and T. Treutner, "Genetic algorithms for energy efficient virtualized data centers," in *Proc. of the 8th International Conference on Network and Service Management (CNSM)*, Oct. 2012, pp. 422 –429.

[14] G. Wu, M. Tang, Y.-C. Tian, and W. Li, "Energy-efficient virtual machine placement in data centers by genetic algorithm," in *Proc. of the 19th international conference on Neural Information Processing - Volume Part III (ICONIP'12)*, Doha, Qatar, 2012, pp. 315–323.

[15] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2012, pp. 1245 –1250.

[16] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. of the 4th USENIX conference on Networked systems design and implementation (NSDI'07)*, Cambridge, MA, April 11-13, 2007.

[17] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proc. of the 2008 ACM/IEEE conference on Supercomputing (SC'08)*, Austin, Texas, Nov. 15-21, 2008, pp. 53:1–53:12.

[18] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder, "Validating Heuristics for Virtual Machines Consolidation," Microsoft, Tech. Rep. MSR-TR-2011-9, Sept. 2011. [Online]. Available: http://research.microsoft.com/pubs/144571/virtualization.pdf

[19] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types," in *Proc. of the 8th USENIX conference on Networked Systems Design and Implementation (NSDI'11)*, Boston, MA, March 30 - April 1, 2011.

[20] A. Caprara and P. Toth, "Lower bounds and algorithms for the 2-dimensional vector packing problem," *Discrete Applied Mathematics*, vol. 111, no. 3, pp. 231–262, Aug. 2001.