Survey on Virtual Machine Allocation in Cloud Infrastructure

Ashwini Ukarde
Department of Computer Science
CSULB
ashwiniganesh.ukarde@student.csulb.edu

Neha Bhoi
Department of Computer Science
CSULB
Neha.Bhoi@student.csulb.edu

Saishree Jayakumar
Department of Computer Science
CSULB
saishree.jayakumar@student.csulb.edu

***Abstract*—Cloud computing is a great boom to today's businesses especially in this pandemic providing highly available and most reliable services independent of the geography through internet. With increased usage in recent times, it strives hard to continue its promising nature of cloud services. In cloud computing, virtual machine allocation is a critical research area which aids in allocating the virtual resources on top of physical machines. Virtual machine (VM) placement algorithms deal with allocating different VMs onto the physical machines in an efficient manner with optimal load balancing among the resources available. This paper proposes a profound survey on various VM allocation policies and focuses primarily on the parameters including energy efficiency, load balancing, resource utilization, time efficiency and SLA time.**

***Index Terms*—Cloud computing, Bin packing, Genetic algorithm, Virtualization, Hybrid genetic algorithm, Resource allocation, Time and energy efficiency, Service Level Agreement, Server consolidation.**

I. Introduction

Cloud computing provides delivery of on-demand services including software applications, servers, networking, databases, storage hosted by cloud service providers through the internet. Cloud services offer faster, reliable and flexible resources at scale. The ease of use provided by cloud providers like Amazon, Google, Azure to spin up compute nodes on demand has made it easy for millions of users to do their day to day activities especially in this COVID era. With the employment of the cloud services, businesses are able to pay for only their usage which tremendously reduces the operation and infrastructure cost. The contribution of virtualization techniques and cloud services lead to the establishment of cloud datacenters. Broadly, the cloud services can be classified into three models differing on their way and level of the datacenters as: Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS). These are known as "stack", as they are built on top of one another. The merits of cloud services include instant on-demand services, cost, productivity, reliability, scalability etc. Virtual machine allocation in cloud infrastructure is nothing but a mapping between virtual host and physical machine under the availability of actual physical resources in the data centers. Today there is an increasing demand for services and resources on cloud platforms. In order to meet this increasing demand in a timely manner we are in need of efficient and optimal Virtual Machine allocation strategies which takes into account varying parameters. Cloud service providers face few major challenges including resource underutilization, high energy consumption, imbalanced load distribution, higher SLA time, increased number of VM migrations and encounter difficulty in meeting the demands of their customers. Several algorithms have been proposed to overcome these issues and offer ceaseless services to the end users. This paper presents a deep survey on 6 different algorithms optimizing for the above specified factors.

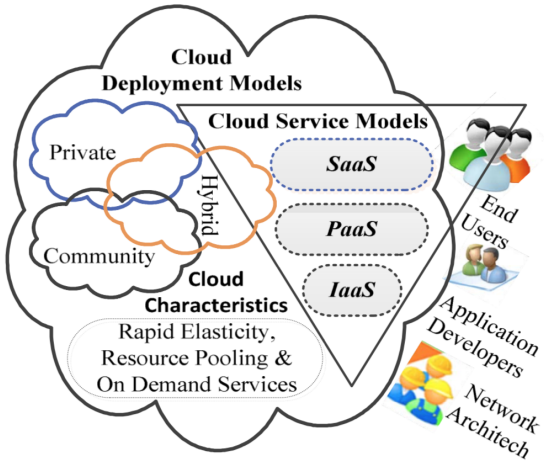


Fig. 1. Model of Cloud Computing

The paper is organized as follows. Section 2 provides a detailed survey on VM allocation algorithms. Followed by Conclusion in Section 3.

II. Comprehensive Analysis of VM Allocation

A. *Interior Search Algorithm(6)*

In the recent era, cloud computing is transforming the way users are consuming on-demand computational resources over internet. The widespread acceptance of cloud and its rapid growth led to the development of cloud data centers. With the increased demand of resources and services, there is a high energy consumption by the data centers thereby the infrastructure cost turns out to be both expensive and at

the same time have become unfavourable to the environment with the huge carbon emission. Thus, there is an urge for an efficient Virtual Machine allocation technique that could overcome the high dissipation of energy caused by poor resource utilization. This work proposes an Energy Efficient Interior Search Algorithm for efficient Virtual Machine allocation. Interior Search Algorithm is a global optimization algorithm used to solve multi dimensional optimization problems. It is inspired by interior design and decoration and is a meta heuristic algorithm. It uses a stochastic random search. The goal of any optimization problem is to find a global optimum. For this purpose the objective function of ISA maximizes for an optimal placement of Virtual Machine minimizing the energy consumption and maximizing the available resource utilization.

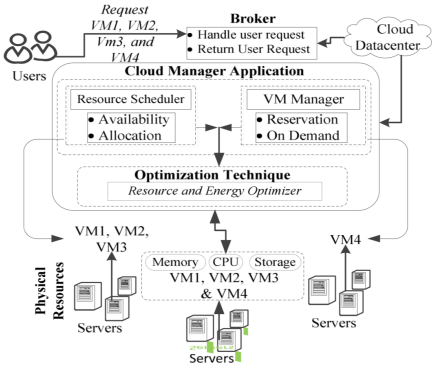


Fig. 2. Proposed System Architecture of ISA

*1) **Architecture**:* Every application in the cloud datacenter has different requirements and cloud datacenters need to manage the demands of the user applications, generating the relevant challenge of on demand provisioning of resources and allocations of varying workloads. The Figure 2 shows an outline of the cloud infrastructure on how the user requests are handled by the broker and eventually to the data centers. At first, the user request will hit the broker. The broker upon analyzing the request will return the assignment back to the user. This assignment will be on the basis on the performance and capacity requirement in the datacenters. Upon the broker's request reached by the cloud manager, it will make an informed decision based on the comparison between virtual machine manager and resource scheduler modules. On the basis of the availability of system, the cloud manager accepts the user requests. But this technique still encounters the difficulty in reducing greater energy utilization. But the proposed approach has undertaken this problem and attempted to apply a new technique of ISA for better results.

*2) **Interior Search Algorithm**:* The idea is proposed by Gantomi and used by various researchers to solve complex multidimensional optimization problems. The performance of ISA has been established using the standard benchmark techniques. It is believed that the approach has the flexibility, capability and efficiency to resolve various NP-hard problems. The elements of Interior Search Algorithm include mirror and composition group. The components of the composition vary in order to acquire greater enticing solutions. On the other hand, the components of mirror is placed between the competent element and mirror element to obtain improvised view. Every optimization technique will have a hand full of tuning parameters. But Interior Search Algorithm utilizes one hyper-parameter (α) for better tuning. The possible values that the hyper parameter could take varies between 0 and 1. The experiment results are being compared with other constraint optimization problems.

*3) **Energy Efficient Resource Allocation in ISA**:* Identifying appropriate optimal solutions in planning, allocating and assigning available resources in cloud data centers is the end objective of this allocation mechanism. This resource allocation problem is also connected to other scheduling issues. A data center would consist of a number of servers which is also known as PM. Each server(PM) has an allocatable quantity of CPU, Memory and Storage connected with other components. Users or clients upon requiring the cloud services would send a request to the cloud provider through a broker. These requests are handled by the datacenter manager which analyzes the request and allocates VMs according to the request of the user. Depending on the application requirement, VMs can be different or identical. There can be different classes of VMs provided by the cloud providers for their clients where these VM differ with respect to memory, CPU and storage of resources. Eventually, every VM is mapped to the PM of the data centers. The following are the assumptions for processing the request of the users by the system model. It includes,

- Cloudlet - Actual tasks that runs on VM
- Host - With the aid of Hypervisor it manages the VMs
- Datacenter - Physical Machines that runs VM
- Datacenter Broker - Agent for processing requests
- Every VM must be hosted by the host and at the same time VM and host are independent

The following is the mathematical formulation for resource allocation,

$$\sum_{x=1}^{Z,n}(PC_i + PD_i + PM_i) \times T_i \longrightarrow U_i^j \qquad (1)$$

In the cloud datacenter, there are m virtual resources V = (V_1, V_2, ... , V_m) and n physical resources. The capacity of each physical resource is represented as, P = (P_{d1}, P_{d2}, P_{d3}, ... , P_{dn}), P = (P_{c1}, P_{c2}, P_{c3}, ... , P_{cn}) and P = (P_{m1}, P_{m2}, P_{m3}, ... , P_{mn}). These physical resources are mapped onto the cloud service users U = (U_1, U_2, U_3, ..., U_n) in a manner that the objective function F = (F_1, F_2, ..., F_z) are maximized concerned with Energy Consumption and Resource Utilization.

Minimizing Energy Consumption: First and foremost goal of ISA is to minimize the energy consumption of the data centers. It is achieved based on the strength requirement for

the cloudlet execution and the user request on the cloud environment. It is calculated by the total electricity power consumed to run every host in the data centers. It is formulated as,

$$Energy\,Cons = \sum_{resource^i} \int_{Str_Ttime}^{Fnh_Ttime} E_i(F,T) \qquad (2)$$

The equation expresses the energy consumption of any resource i at the time T with placement of F. E_i represents the energy consumption of resource *i* from the start of the host till the finish time.

Maximizing Resource Utilization: The resource utilization model is a measure of actual utilization of resources by the data centers. The second objective of ISA is to maximize this resource utilization. Henceforth, the cloud manager has to make an optimal assignment to enhance the efficiency of allocation between the user request and the available resources at that particular time. The resource utilization is calculated as,

$$Resource\,Utiliz = \frac{\sum_{resource^i} Execution_{time}}{Makespan\,or\,Maxtask^i(Execution_{time})} \qquad (3)$$

Makespan represents the highest completion time once the resources are allocated to the users and the execution time represents the difference between the start time and finish time.

4) ***Performance Evaluation****:* The experimental analysis of the proposed algorithm is performed using CloudSim. The following table showcase the parameter setting of the scheduling techniques,

TABLE 1. EXPERIMENTAL PARAMETER SETTING

S/NO	Parameter Setting				
	Technique	*No OF Host*	*No of VMs*	*No of Users*	*VMM*
1	EE-IS	25	180	10	Xen
2	GA	25	180	10	Xen
3	BFD	25	180	10	Xen

The algorithm is implemented in the following system requirements,

- 8 Gigabyte memory
- Intel i5 processor
- 2GHz processor speed
- 1 Terabyte hard disk drive

In the analysis, the proposed approach has been compared with other existing traditional algorithms including Best fit decreasing and Genetic Algorithms on the basis of total energy consumption and resource utilization. It is evident from the Figure 3 that the energy consumption of BFD and GA is 90-95% whereas ISA consumes only around 65% of energy. On an average, EE-ISA saves around 30% of energy and at the same time the resource utilization is tremendously improved.

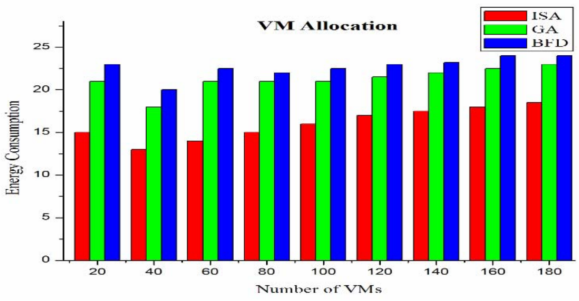


Fig. 3. VM Allocation for EE-IS, GA and BFD

One of the challenges in ISA is that, being an optimization algorithm it runs for multiple iterations to converge to a global optimum affecting the response time for Virtual Machine creation thereby affecting SLA for users. The particular work doesn't evaluate the possible solutions for this challenge. One possible solution based on other research papers could be to mitigate this by parallelizing the stochastic random search to reduce the time complexity with fewer margins of errors. It can be conceived that taking the same approach to ISA will help reduce the latency.

From experimental results it is evident that ISA outperforms GA and BFD and the promising results of ISA shows that the approach could be adopted in real datacenters. This proposed approach also tremendously improve resource utilization.

B. K-Means Algorithm

One of the daunting fields of virtual machines is resource distribution in cloud computing. It is the most critical challenge that cloud computing has. The K-means algorithm for virtual machine allocation is one way to maximize resource utilization and allocation. The K-Means algorithms use the technique of clustering to effectively allocate virtual machines. Compared to the current FIFO algorithm, the result obtained demonstrates that the K-means algorithm is better for CPU utilization use and has lower idle time. In virtual machine allocation, it also does load balancing.

1) ***K-Means algorithm****:* In the K-Means algorithm, clustering technology is used. Large datasets are broken up and groups with similar characteristics are formed. The algorithm is a simple clustering method, starting from the Point selection as the original centroid K, where K is the number of user-specified clusters. Every point is assigned to the nearest centroid and each group of points is called a cluster. Depending upon the number of points in the cluster, centroid of the cluster is updated. The process of updating the centroid is continued till the time the centroid remains unchanged. This paper uses the Euclidean distance formula to calculate the distance between centroid and points.

$$\delta(x,y) = \left(\sum_{i=1}^{d} (x_i\text{-}y_i)^2 \right)^{1/2} \qquad (4)$$

After distance calculation is completed then any point performs assignment at the nearest centroid. Further calculations on each centroid cluster can be done with the below equation.

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x \tag{5}$$

*2) **CloudSim – Simulation Framework**:* The simulation using the CloudSim platform is seen in this survey. CloudSim has classes such as brokers, CIS (Cloud Information Service), a virtual machine, cloudlet (cloud computing job or task), datacenter and hosts that act as a simulation of cloud computing. The diagram below illustrates the overall configuration of the device.

*3) **System Design**:* In this paper, the number of clusters is determined dynamically depending upon the number of datacenters and hosts. Data center characteristic information collected via the Cloud Information Service (CIS) that conducts a cloud provider query. After that the broker decides the K cluster number.

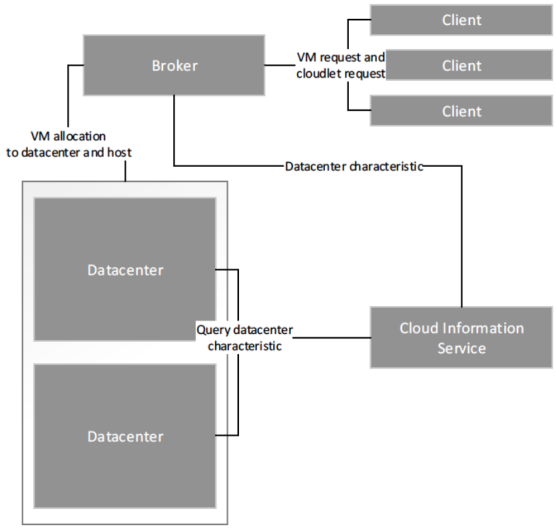


Fig. 4. System Design

The CloudSim framework built the datacenter and brokers in the above system design from the initialization package CloudSim, then CIS (Cloud Information Service). The broker executes the request for the virtual machine and then places it in the datacenter on the hosts and sends the cloudlet to each virtual machine, and the simulation runs and finishes with the performance of the simulation, such as the amount of time, the number of cloudlets, the number of virtual machines, the number of hosts and the number of the datacenter.

*4) **Cloudlet Scheduler**:* This paper uses the Cloudlet Shared Space Scheduler Cloud computing scheduler that has been provided by CloudSim, and in the CloudletSchedulerSpaceShared class. Cloudlet1 must be completed first before executing on cloudlet2 in the cloudlet space scheduler phase.

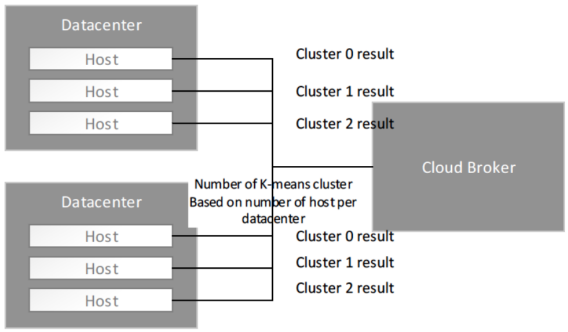


Fig. 5. Clustering

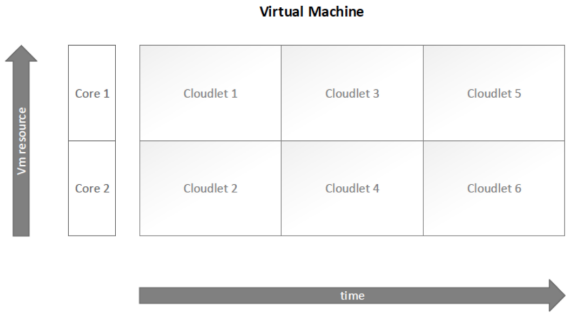


Fig. 6. Cloudlet Scheduler

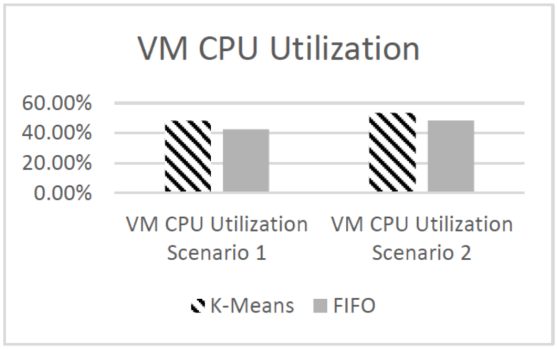


Fig. 7. CPU Utilization

*5) **Experimental Results**:* The experiment result with K-means allocation method gives 48.1993 percent CPU consumption and for the FIFO method is on the other hand, virtual machine CPU consumption is 42.539 percent.

With K-means, the gross idle time is 132614.7 seconds, and when that number is minimal, relative to the 160896.66 idle FIFO process, Just seconds. This is because the allocated number of virtual machines with K-means is less than the FIFO process, but with the same running time, so the allocation of virtual machines with K-means is better than the approach to FIFO.

In the K-means process, the virtual machine has less idle time and is distributed in each data center in a balanced way (load balance) according to the degree of specification

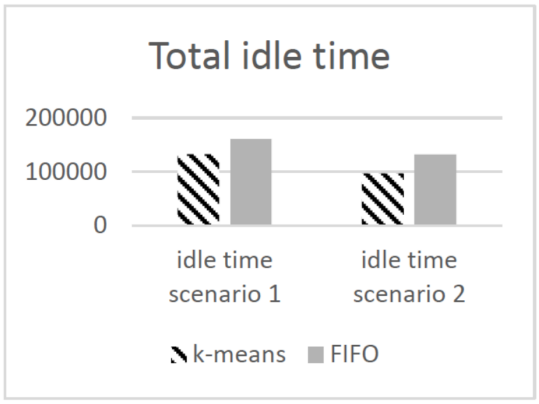


Fig. 8. Idle Time

similarity within the virtual machine. Every data center will be assigned a job on the default cloudlet scheduler used by CloudSim according to the number of virtual machines in accordance with the number of virtual machines without the computing capabilities of that virtual machine being measured.

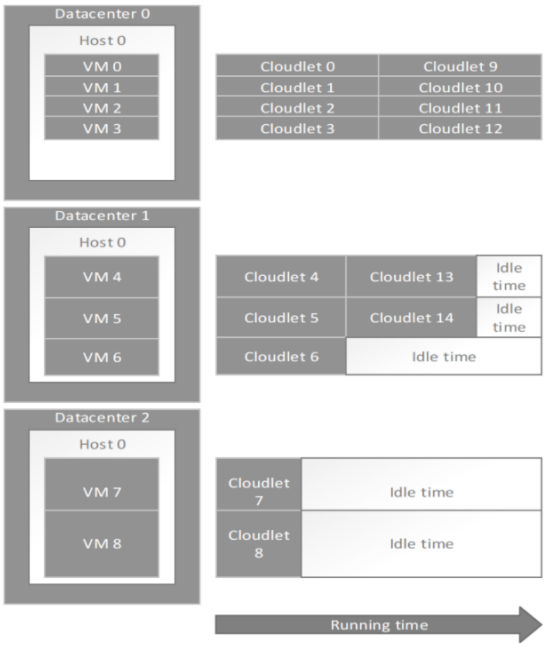


Fig. 9. VM Allocation using K-Means

*6) **Analysis**:* The consequence of this study is that the virtual machine allocation approach with the K-means algorithm is easier than the FIFO algorithm in order to boost the use of the virtual machine CPU. The use of the K-means algorithm in the virtual machine allocation scheme will also improve the load balancing of the data center host resource relative to the FIFO algorithm approach and the use of the virtual machine CPU during the K-means algorithm method simulation.

Approach	Solved
Relaxed MBS' [5]	2
VNS [5]	2
Perturbation MBS' & VNS [5]	2
GA [8]	3
BISON [2], [8]	3
Dual Tabu [2], [5]	3
Exon Shuffling GA [11]	8
RGGA	**9**

Fig. 10. RGGA compared with other algorithm

C. *Reordering Grouping Genetic Algorithm (RGGA)(2)*

Reordering grouping genetic algorithm (RGGA) is an extension of native genetic algorithm which focuses on resolution of multi-capacity bin packing problem (MCBPP). This main feature of this algorithm is the characterization of individuals in genetic algorithm. Virtual machine allocation is instance of multi-capacity bin problem where every single bin has more than one capacity as well as more than one weights for every single objects. The main constrain for this problem is that for any assets the total of weights must not exceeds the equivalent capacity.RGGA is based on the idea of first fit ordering which focus on producing practical solutions. It defines below operator:

- **Fitness Function:** The fitness feature is used to measure the fitness of an individual event.
- **Crossover Operator:** The crossover operation in any genetic algorithms focus on incorporating features of people in the community to generate up to the minute generation. This operator will arrange the bins in the order of fitness which will results having closely packed bins at the start than the last bins in the collection.
- **Mutation Operator:** There are three choices provided by mutation operator. In the first choice the mutation operator will take some quantity of bins from the given collection of the solution and reinserted some items to the nominee solution. In Second, couple of items are swapped in the object packing series. In the last choice the element is transferred to different location in the packing series.

1) State of the art:

- **Based on:** Genetic algorithm and bin packing.
- **Resources Considered:** Processor time and RAM used.
- **New Aspect:** Energy efficient, optimal solution, high throughput.
- **Weakness:** Workloads for VMs are deterministic.
- **Performance Better Than:** Exon shuffling GA, first fit and worst fit

2) Experimental Results: They consider the normal steady-state GA with a population size of 75 to conduct the experiment. A crossover operator with a probability of 0.8 was applied. The mutation operator was added to each individual with a probability of 0.1 after crossover. And execute the two-phase experiment as follows.

- **Reduce MCBPP to Conventional Bin Packing Problem:** Here they reduced the MCBPP's number of capacity

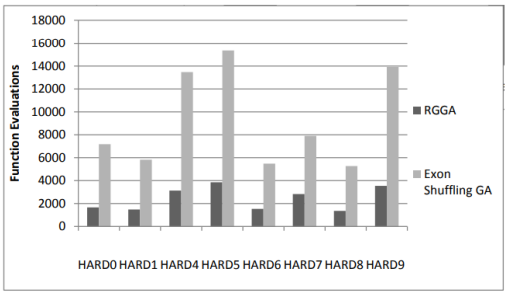


Fig. 11. The average number of function evaluations for the exon shuffling GA are compared with the function evaluations for RGGA

instance	Opt	RGGA Avg.	Exon Shuffling Avg
HARD3	55	55.96	56
HARD4	57	57	57.02
HARD5	56	56	56.06
HARD9	56	56	56.02

Fig. 12. RGGA compared with Exon Shuffling.

Opt	RGGA Found	FFD Found	PP Found
59	60.00	61	61.4
112	113.00	121	117.8
191	192.00	207	196.5
216	217.00	234	223.9
241	242.00	262	250.3
267	268.00	296	277.7
320	321.00	353	331.8
371	372.00	412	384.7
425	426.02	465	439.8
481	482.00	541	499.5

Fig. 13. Summury of experimental results of RGGA on multicapacity problem

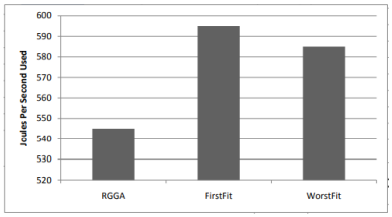


Fig. 14. The amount of energy used in solutions returned by the three different algorithms

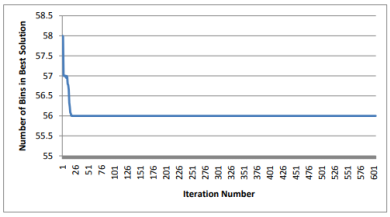


Fig. 15. The average number of bins in the best solution for different iteration numbers for HARD0

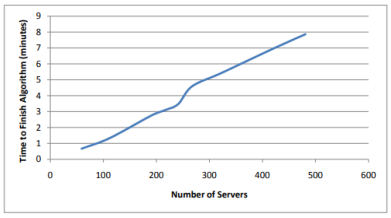


Fig. 16. The amount of time taken to find a solution for varying sizes of Multi-Capacity Bin Packing Problems

to one. They used a collection of instances of the benchmark generated by A. R. and Scholl. Klein. Klein. The third collection has the HARD0-HARD9 mark. There are 10 problems in the data collection, each consisting of 200 items with weights of 20,000 to 35,000 and bin capacities of 100,000. They compared RGGA against the genetic algorithm for Exon shuffling here. Figure 10 shows comparison of several approaches on the same set of 10 benchmark instances of one-dimensional bin packing. 4% of the time, RGGA solves HARD3 to optimality, although the genetic algorithm of exon shuffling never solves it. Figher 11 shows RGGA performs significantly better than exon shuffling.

- **MCBPPs:** A collection of MCBPPs has been built to model the number of VMs usually found on servers.A test set of MCBPPs with close to 9-13 items per bin was produced. According to two different tools, they packed each process: processor time and used RAM. In addition, the loads allocated to VMs are such that nine to thirteen VMs will fit on a physical server. They produced 10 separate instances of two-capacity problems of varying size. In each problem case, RGGA was executed 50 times. Figure 13 shows that in every run, except for one run when the optimal number of bins was 425, RGGA was always one bin away from the optimal packaging. RGGA substantially outperforms the heuristic first fit decreasing (FFD).
- **RGGA For Virtual Machine Deployments:** Here, they evaluated RGGA against the first fit and worst fit algorithms. Artificial loads on VMs were generated and the total amount of energy used by all servers was observed. Figure 13 RGGA uses 8.4% less energy than first fit and 6.84% less energy than worst fit.

3) Time to Run RGGA: Figure 15 shows the average number of bins in the population when running RGGA on HARD0, with different iteration numbers.In less than thirty iterations, RGGA finds an optimal solution. After 100 iterations for different sizes of multi-capacity bin packing problems, they benchmarked the time RGGA required to find a solution. As the problem size increases, Figure 16 shows linear RGGA scales within this range.

D. *A Novel Family Genetic approach for VM Allocation(7)*

The concept of Virtualization associates the creating and deploying VMs onto the physical host where the host machine should be compliant with the specific requirements of VM including CPU, memory, storage etc. The most challenging factor that needs to be considered is the provisioning of VMs to host within the realm of cloud infrastructure since these virtual machines are reliant on the resources and efficiency of the host machines for them to operate. An inappropriate allocation of virtual machines could result in unwarranted results including under utilization / over utilization of few physical machines(load imbalance) and could even affect the credibility of Cloud Service providers. Realizing the impact it creates, several methodologies have been proposed to solve VM placement. One such prominent approach for solving the VM allocation problem is the Genetic Algorithms which is inspired by the mechanism of natural selection. But most of the traditional Genetic Algorithms undergo the limitations of high processing times and premature convergence. Premature convergence is a common issue in an optimization problem where the population converges too early and is not able to generate better offspring that outperforms their parents. Subsequently, it results in sub-optimal results. Another prevalent hindrance of any traditional Genetic algorithm is the duration of the processing time. Since Genetic Algorithm works in terms of producing various generations (need to work with various generations of parents to produce better offspring), it consumes high processing time before bringing forth optimal results. The evolutionary Novel Family Genetic Algorithm proposed by this work shows an efficient technique which strives to overcome these limitations of traditional Genetic Algorithm.

*1) **Problem Statement**:* The Virtual Machine allocation is a multi-objective optimization problem where the goal is finding an optimal allocation i.e a mapping between a VM to host. Consider a system with a set of host machines "m" and virtual machines "n" where individual VM is represented by v_i and host machine is represented by p_j with a single decision variable y_{ij} which could take the value of 1 if the i^{th} VM is allocated to j^{th} host and 0 otherwise. The set of all VMs and host machines are denoted by P and V respectively. Also individual host machine is represented in the form of vector:

$$p_j = (id_j,\ cpu_j,\ mem_j,\ bw_j) \tag{6}$$

where id_j denotes the identification number for the host, CPU_j denotes the processing power of the host, mem_j denotes the memory capacity of the host, bw_j denotes the bandwidth capacity that the host could support.

Similarly, each VM is also represented in the form of vector:

$$v_i = (id_i,\ cpu_i,\ mem_i,\ bw_i) \tag{7}$$

where id_i provides the identification number for the Virtual Machine, CPU_i represents the processing power that the VM requires, mem_i represents the memory requirement of the VM and bw_i represents the bandwidth requested by VM.

The problem could be defined as the placement of set of VMs onto a set of PMs wherein the resource utilization will be maximized.

The objective of maximization of the physical resource utilization is achieved by the decomposing into three objectives:

$$Maximize\left(\frac{v_i^{cpu}}{p_j^{cpu}}, \frac{v_i^{mem}}{p_j^{mem}}, \frac{v_i^{bw}}{p_j^{bw}}\right) \tag{8}$$

Under the constraints of,

$$\begin{aligned} \forall i \sum_{j=1}^{m} y_{ij} &= 1 \\ \forall j \sum_{i=1}^{n} y_{ij}.v_i^{cpu} &\le p_j^{cpu} \\ \forall j \sum_{i=1}^{n} y_{ij}.v_i^{mem} &\le p_j^{mem} \\ \forall j \sum_{i=1}^{n} y_{ij}.v_i^{bw} &\le p_j^{bw} \end{aligned} \tag{9}$$

These constraints ensure the following,

- Even though multiple VM could be mapped to a single host, a single VM could be allocated to only one host.
- It makes sure that the load on the host is within its capacity
- It is possible to define the lower and upper threshold for the utilization on the host machine.

The utilization for i^{th} VM in the j^{th} host machine is calculated as,

$$p_u^{j(i)} = \begin{cases} \left(\frac{v_i^{cpu}}{p_j^{cpu}}.\frac{v_i^{mem}}{p_j^{mem}}.\frac{v_i^{bw}}{p_j^{bw}}\right) * 100, & \text{if } y_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Also, the total utilization of host can be computed as the sum of the utilizations of all the VMs that are placed in a single host,

$$p_u^j = \sum_{i=1}^{n} p_u^{j(i)} \tag{11}$$

*2) **Proposed System**:* The Family Genetic Algorithm (FGA) has been proposed to overcome the deficiency of traditional Genetic Algorithms by analyzing the influence of population size and individual's space on the operators as shown in the Figure 17. The experiments have been conducted by integrating the FGA with the Cloud Sim. Cloud Sim has data centers that consist of hosts with each host consisting of one or more Processing Elements(PE). VMs that are functioning on the hosts run a few cloudlets. Cloudlets represent User jobs with varying requirements. FGA takes the set of hosts and VMs as input and produces an optimal mapping. It is achieved by dividing and parallelizing the processing among several families.

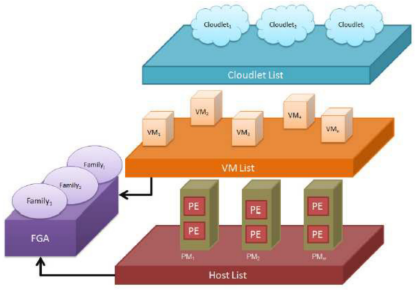


Fig. 17. Proposed System Architecture in CloudSim

One of the major problems of traditional Genetic Algorithms is premature convergence. The most contributing factor influencing this issue is the mutation operator. By utilizing the self-adjusting mutation operator, FGA strives to minimize the likelihood of premature convergence. In traditional GA, the probability of mutation is static, it is fixed at the start of the algorithm and is maintained as a constant for the complete execution. Being a variant of GA, FGA attempts to vary the mutation rate thereby making the mutation probability dynamic using a parameter known as Population Differentia.

Population Differentia is the ratio used to specify the rate at which the different individuals differ from each other. It is the parameter that guides the probability of mutation. The premature convergence is prevented with the usage of self adjusting mutation factor. For instance, the calculation of degree at which the two individuals differ, say "A" and "B" is given by:

$$d(A,B) = \sum_{j=0}^{l-1} A_j \oplus B_j \tag{12}$$

l represent the length of chromosome

Total rate at which each individual differ from the rest of the population can be defined as:

$$Population\ Differentia = \sum_{i=0}^{N}\sum_{j=0}^{N} \frac{d(A_i,B_j)}{(N-1).(N-1).l} * 100 \tag{13}$$

N represents the population size.

In traditional GA, the entire population is taken as whole and applied with different operators including selection, crossover and mutation at once across all generations. It has been proved by the researchers that these steps consume the most time in the algorithm. Being a variant of GA, FGA works by dividing the entire population into various families and performing all the specified operations among multiple families in parallel which certainly enhance the speed of the Family Genetic Algorithm thereby reducing total runtime. The approach of constructing families was first proposed by Jianhua et al. which considers the solution from neighbours. But our problem of VM placement does not formulate any neighbours. Hence the construction of families is achieved through simple mutation. The derived chromosomes although vary slightly are placed in the same family. Further reduction in the processing time is achieved by discounting the families which fails to yield better individuals. Every family is processed "k" times and the family will be destroyed if they didn't produce a better individual till then and proceed with the next family. But if the algorithm generates atleast one better individual in the current family then the processing of that particular family continues for "W" iterations. The value of k and W are tuned through the experimental results.

The quality of each individual in the population is assessed by calculating the fitness value related to it. In traditional Genetic Algorithms the value of fitness function is an objective function that is under consideration. In FGA, our objective is to maximize the physical resource utilization and the fitness function is outlined based on it. Additional provision has been employed in FGA where every chromosome should satisfy the specified constraints. To achieve it, a separate feasibility check is performed for the chromosomes. If a chromosome is identified as infeasible, then the measures will be taken to transform them into a feasible one.

*3) **Experimental Results and Analysis**:* The performance analysis of the algorithm was done using the CloudSim toolkit. The proposed FGA approach is compared with various existing methods.

On analyzing the Figure 18 and Figure 19, it is apparent that the allocation policy uses only the required hosts and allocate all the VMs thereby reducing the total number of hosts that are being used. The hosts that are not used in mapping could be switched off furthermore reducing the energy consumption. In recent days, there is a greater concern on increased energy consumption since it has a direct impact on the environment. With the proposed approach, the number of hosts in operation is relatively lower than other existing policies contributing to the reduced energy consumption. Hence FGA is not only optimized for higher physical resource utilization, but also uses power aware VM allocation approach. From Figure 20, it is proven from the experiment that FGA optimizes for energy consumption for about 4-9 times compared with other existing algorithms.

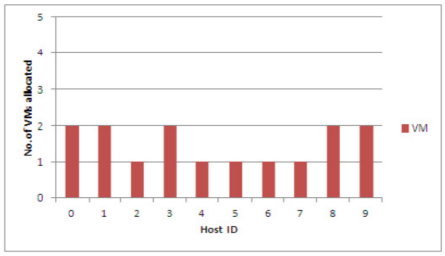


Fig. 18. Initial VM Allocation

Given an allocation policy, if resultant allocation does not meet the resource requirements of a VM, then a situation arises to move the VM from one physical resource to another suitable

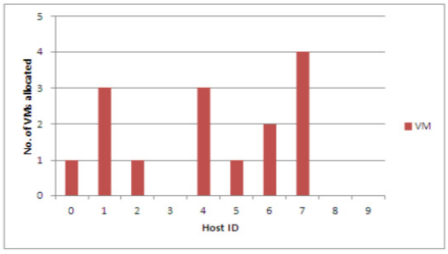


Fig. 19. VM Allocation using Proposed FGA Approach

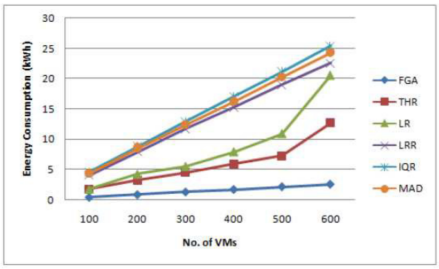


Fig. 20. Comparison of Energy consumption between existing and proposed approach

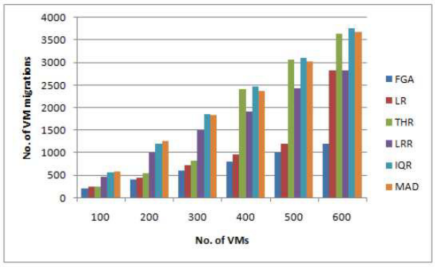


Fig. 21. Comparison of number of VM Migrations between existing and proposed approach

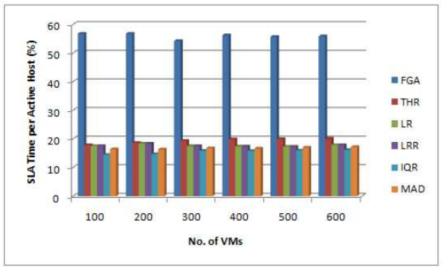


Fig. 22. Comparison of SLA time per active host between existing and proposed approach

host. This process is known as VM Migration. Usually these VM migrations experience overhead on the system and always it is preferred to keep such VM Migrations at minimum. Experiments have been formed to analyze the number of VM migrations required by existing and the proposed approach. It is observed from the Figure 21 that the number of migrations in FGA is 3-5 times lesser than other existing approaches.

The Service Level Agreement(SLA) is another factor in determining the efficiency of VM Allocation algorithms. It is an agreement between the cloud service provider and the user where the cloud provider consents to meet specific conditions of the user. One of the major objectives of any cloud provider is to obtain higher SLA. It is factored by SLA time per host depending on the allocation policy. It provides the time in terms of percentage where every host follows the SLA. From the Figure 22, it is evident that the proposed approach has a higher SLA time per active host which guarantees overall higher SLA level. SLA of FGA is at least 3 times faster than other algorithms.

In essence, Family Genetic Algorithm is optimized to achieve higher resource utilization, minimized energy consumption, reduced VM migrations and increased SLA time per slot.

Even though FGA has a lot of advantages, it suffers from few disadvantages. It needs multiple parameters to tune including k, W and number of families to be constructed. It has less information about how the mutation and crossover is performed. Although it is an adaptive GA, FGA takes very small steps to converge during last few iterations. One possible solution for slow convergence at the last iterations is to combine FGA with other optimization methods like Hill Climbing to improvise the problem of convergence based on other research papers. It would be interesting to combine both these algorithms to see the performance in real datacenters.

From the experimental results it is evident that FGA outperforms traditional Genetic Algorithms, THR, LR, LRR, IQR and MAD and the promising results of FGA shows that the approach could be efficiently adopted in real datacenters. Since FGA tremendously reduces energy consumption, it could also be employed in green datacenters.

E. Bin packing with variable item size (VISBP)

Resource provisioning is one of the essential aspects of allocating virtual machines. During certain times of the day, resource requirements can rise or decrease. The allocation of a fixed number of resources can lead to resource wastage or resource under-use. Determining the number of resources for applications for on-demand processing is uncertain. Holding the system idle would also lead to a waste of energy. For load balancing and energy saving purposes, this algorithm uses live migration methods. When the usage of resources is at the highest, few virtual machines can be relocated to decrease the load on the server, whereas it can be placed in sleep mode to conserve energy as the load is very low or the system is

idle. Energy is one of the essential costs of running virtual machines.

This algorithm utilizes techniques for live migration to dynamically allocate resources based on resource requirements and also promotes green computing as it optimizes the number of servers used.

*1) **System Architecture**:* Gray color highlights the components of this system. Each virtual machine (VM) runs on the top of virtual machine monitor (VMM). This VMM is placed on a physical machine (PM) and multiple virtual machines runs on single physical machine. The VMM gathers resource use statistics on the required PM, such as CPU utilization, memory use, network sending and receiving, etc. VM scheduler uses these statistics for load balancing and green computing. This is done using allocating virtual machines to physical machines.

The VM scheduler is frequently invoked and receives VM resource demand history, PM capability and load history and current VM assignment to PMs. It has two parts, first part: the load predictor predicts the resource requirements. Second part: Scheduling algorithm helps to optimize the VM layout so that the demands for resources are met and resource waste is reduced.

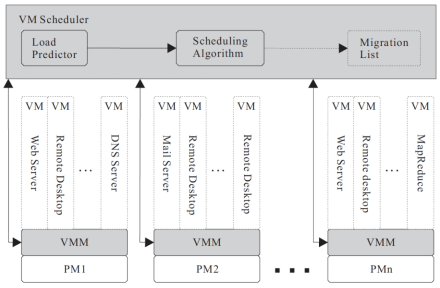


Fig. 23. System Architecture

If the scheduler is invoked, the load predictor calculates the resource requirements and loads the PM, accordingly, using previous estimates for the resources for this function. The figures are expressed by the number of CPU ticks, the sum of memory and network I/O bandwidth. With these figures, it will check if the system has required resources. If all the required resources are available, then VMM can locally adjust the resources accordingly. The scheduling algorithm migrates the virtual machines if there is overload or it will put the system in sleep mode if the system is idle for a long time. This scheduler algorithm produces a VM migration list which is then forwarded to the VMM. Then the live migration technique is used to adjust the resources.

The exponentially weighted moving average (EWMA) load predictor is used in this algorithm:

$$E(t) = \alpha * E(t\text{-}1) + (1\text{-}\alpha) * O(t), 0 \leq \alpha \leq 1 \qquad (14)$$

Where, E(t) and O(t) represents a tradeoff between stability and responsiveness, approximate load at time t and α . The algorithm's consistency and efficiency are strongly dependent on this predictor algorithm.

*2) **One Dimensional VISBP**:* This algorithm separates objects according to their sizes into four forms. Insert item involves inserting items in any bin. Change (existing item, item) is an old item already packaged and has become a new item (differ only in size) (differ only in size)

Number of Movements in Worst Case

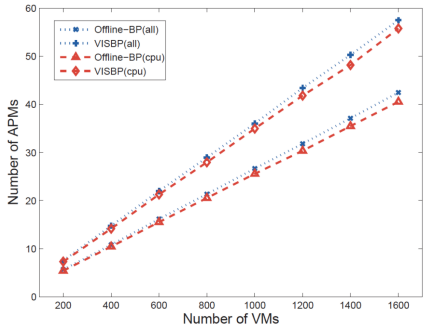
Old \ New	B-item	L-item	S-item	T-item
B-item	0	3	1	1
L-item	2	3	3	3
S-item	1	4	0	2
T-item	5	7	6	5

*3) **Multi-dimensional VISBP**:* By breaking down objects according to their greatest dimensions, we expand VISBP to multi-dimensional.

Input: $L_d = (v_1, v_2, ..., v_n)$
where $v_i = (v_i^1, v_i^2, \ldots, v_i^d)$ and $0 \leq v_i^j \leq 1$
$(i = 1, 2, \ldots, n \; and \; j = 1, 2, \ldots, d)$
Output :

- Positive integer $A_d(L_d)$: Number of total bins used.The bins are ranged from 1 to $A_d(L_d)$.
- $O_d = (1, b_1), (2, b_2), \ldots , (n, b_n)$, where b_i is the index of the bin for item i ($1 \leq b_i \leq A_d(L_d)$.

*4) **Experimental Results**:* Below figure shows the AMP comparison of offline BP and VISBP



*5) **Analysis**:*

- VISBP maintains an even distribution of loads close to the offline version BP, and significantly better than of BG and VectorDot algorithms.
- Around 73 per cent of total PMs are used by our VISBP algorithm, Far less than 98
- In order to receive item VISBP uses unused bins and have some spare space to handle fluctuations in load unlike offline bin packing.
- VISBP produces far less cold spots than the algorithms BG and VectorDot.

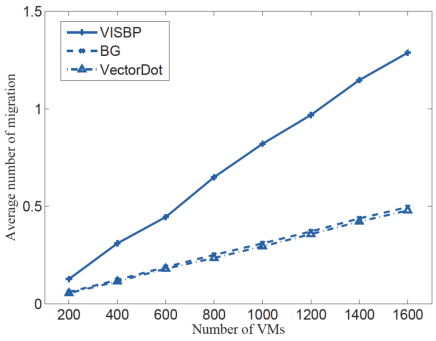


Fig. 24. Experimental Results

F. *Hybrid Genetic Algorithm (HGA) (1)*

HGA is nothing but the extension of the genetic algorithm. The extension consists of a repairing procedure for impractical solutions as well as a local optimization procedure. Generally, any VM allocation algorithm do not take account of energy consumption by communication network in datacentres. The fact that utilization of energy by communication network is not negligible. The hybrid genetic algorithm is an algorithm for VM positioning that takes into account energy utilization for both PMs and the transmission network. It consist of below algorithms:

- **The infeasible solution repairing procedure:** The fundamental concept behind the repairing procedure for impractical solutions is to rearrange the virtual machines which are currently assigned to the given physical machine one after the other prior to the processor breaches CPU and key memory limits on the physical machine are resolved.
- **The local optimization procedure:** Provided the practical virtual machine allocation solution, the local optimization approach uses a heuristic algorithm. This will be worked on reducing the number of physical machines in the given virtual machine allocation solution and will eventually bring down the physical machine's total energy utilization.

1) State of the art:

- **Based on:** Genetic Algorithm and Heuristic Algorithm.
- **Resources Considered:** CPU, Memory and Network.
- **New Aspect:** Energy efficiency of communication network and Optimal Solution.
- **Weakness:** A data structure is needed to reflect a VM placement. The integrated repair and local op-timization process can lead to an increase in the workload of computation.
- **Performance Better Than:** Original Genetic Algorithm

2) Experimental Results: To carry out the experiment, they established a Java project to create virtual machine allocation problems of various configuration randomly. This project will be used to create two sets of test problems. First set will have set of 20 physical machines and the amount of virtual machine will varied. The second set will have fixed set of virtual machines with 20 virtual machines while the amount of physical machine will be varied. The detailed configuration of these set will be shown in the figure 25. They created the native genetic algorithm and HGA in java programming language. This project is used to work out each of the experimental problems.

- **Performance:** In order to evaluate the outcome of the experiment, here they have analysed the excellence of the results generated by the HGA with the native genetic algorithm's results. This analysis is carried out by tak-

Group 1			Group 2		
Test Prob. Set (ID)	VM (#)	PM(#)	Test Prob. Set (ID)	VM (#)	PM(#)
1	40	20	10	80	17
2	50	20	11	80	20
3	60	20	12	80	23
4	70	20	13	80	26
5	80	20	14	80	29
6	90	20	15	80	32
7	100	20	16	80	35
8	110	20	17	80	38
9	120	20	18	80	41

Fig. 25. Characteristics of the test problems

Test	GA		HGA		t-Test Results		
Prob. Set	Energy	Variance	Energy	Variance	t Stat	P(T ¡= t)	t Crit
1	4011	348237	5748	242036	-14.50	7.97E-15	2.05
2	5121	59691	7329	40430	-12.97	1.33E-13	2.05
3	6013	366316	8657	761945	-16.19	4.64F-16	2.05
4	7067	6303833	10456	62465	-16.67	2.15E-16	2.05
5	8375	1130730	12121	1848443	-12.87	1.63E-13	2.05
6	9383	715127	14705	7132933	-11.00	7.21E-12	2.05
7	10686	1307446	18509	12781065	-13.32	6.85E-14	2.05
8	11855	1672803	21134	11319707	-13.12	1.01E-13	2.05
9	13999	1447579	24730	5781698	-22.28	8.63E-20	2.05
10	8171	516275	13306	5986457	-10.86	9.82E-12	2.05
11	8392	1012060	12517	1752620	-19.79	2.18E-18	2.05
12	8282	943118	11898	1277124	-12.93	1.43E-13	2.05
13	8014	877549	11951	931759	-16.32	3.69E-14	2.05
14	7829	918627	11854	686714	-18.48	1.38E-17	2.05
15	7748	829955	11657	789199	-18.26	1.89E-17	2.05
16	8273	1128085	11390	649527	-14.56	7.19E-15	2.05
17	8054	930989	12085	12085	-21.49	2.29E-19	2.05
18	7982	914841	11728	1015671	-13.99	1.97E-14	2.05

Fig. 26. Comparison of the performance of the GA and HGA.

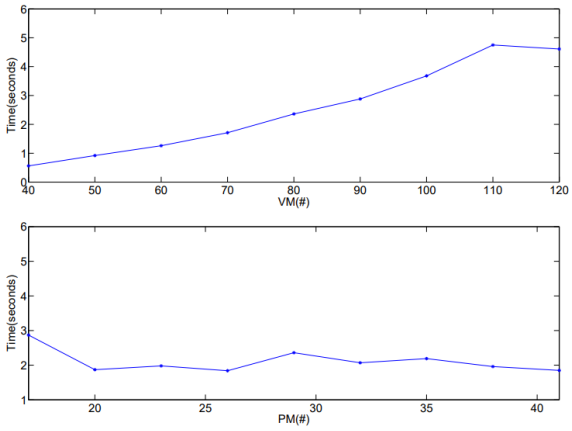


Fig. 27. The scalability of the HGA

Name	Based On	Resource Considered	Strength	Weakness	Performance better than
Interior Search Algorithm	Stochastic random search	Number of available Hosts, VMs and Users	Energy Efficiency and Resource Utilization	Response Time due to multiple iterations need to reach global optimum	Best Fit Decreasing and Genetic Algorithm
Novel Family Genetic Approach for VM Allocation	Genetic Algorithm	CPU, memory and bandwidth	Energy Efficiency, better SLA, less migration	FGA takes very small steps to converge during last few mutations	Traditional GA, Single factor regression algorithm(LR and LRR), IQR, MAD
K Means	K-Means Algorithm	CPU Utilization	Time efficient	NA	FIFO algorithm
Bin Packing with Variable Item Size Algorithm	Bin Packing	CPU, Memory and Network	Hot-Spot mitigation, Load balancing	SLA Violation	VectorDot, Offline BP, Sandpiper
Reordering Grouping Genetic Algorithm	Genetic Algorithm , Bin Packing.	Processor Time , RAM Used.	Energy efficient, Optimal Solution, High throughput	Workloads for VMs are deterministic.	Exon shuffling GA, First Fit and Worst Fit
Hybrid Genetic Algorithm	Genetic Algorithm , Heuristic Algorithm.	CPU, Memory and Network	Energy efficiency of communication net-work and Optimal Solution.	Needs a data structure to represent a placement of VMs. The incorporated repairing and local optimization procedure will result in the increase in the computation workload	Original Genetic Algorithm

Fig. 28. Comparison of State-of-the-art VM Placement Algorithms

ing into account of collection of experimental problems created randomly. For the 30 separate experimental problems, the mean average energy consumption of the HGA from Figure 26 is 27.36 percent-43.90 percent lower than that of the initial GA.

- **Efficiency:** For the purpose of performance evaluation, a collection of randomly created experimental problems are considered. They analyse the operation time of HGA along with the native genetic algorithm. The average HGA operation time decreases by 73.30 percent-88.61 percent for the 30 separate experimental problems.
- **Scalability:** As the magnitude of experimental problem increases, the extensibility of HGA is analyse by examining at how its operational time increases. The shape of operational time graph for the situation mentioned as the quantity of virtual machine rasied from 40 to 120 with incremental of 10 when the quantity of physical machine was set at 20. Also, the shape of operational time graph for the situation mentioned as the quantity of physical machine raised from 17 to 41 with an incremental of 3 where the quantity of virtual machine was set at 80. The figure 27 results that the HGA is scalable.

In conclusion, HGA has greater potential for exploitation and thus produces better solution quality than GA. HGA is stronger than GA for convergence. Unlike all those heuristic algorithms, HGA can provide much better solutions.

III. Conclusion

In this paper, we did a comprehensive analysis on 6 different VM allocation algorithms specifying the various parameters it optimizes for, strengths, drawbacks and even possible mitigative solutions as shown in the Figure 28. Table 1 shows the list of algorithms and parameters it optimizes for. From our analysis we realized that the algorithms have room for improvements. In the future scope we would try to prototype and experiment using the possible solutions that we specified and analyze whether it gives better results.

Parameters	ISA	FGA	KMeans	VISBP	RGGA	HGA
Energy Efficiency	Yes	Yes		Yes	Yes	Yes
Resource Utilization	Yes	Yes	Yes		Yes	Yes
Lower VM Migrations		Yes		Yes		
Better SLA Time		Yes				
Time Efficient			Yes		Yes	Yes
Load Balancing	Yes			Yes		
Hotspot Migration				Yes		

TABLE I
Feature Comparison on Various Algorithms

REFERENCES

[1] Tang, M., Pan, S. A Hybrid Genetic Algorithm for the Energy-Efficient Virtual Machine Placement Problem in Data Centers. Neural Process Lett 41, 211–221 (2015).

[2] D. Wilcox, A. McNabb and K. Seppi, "Solving virtual machine packing with a Reordering Grouping Genetic Algorithm," 2011 IEEE Congress of Evolutionary Computation (CEC), New Orleans, LA, 2011, pp. 362-369, doi: 10.1109/CEC.2011.5949641.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proc. of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05), 2005.

[4] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in Proc. of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07), 2007.

[5] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in Proc. of 30th IEEE International Conference on Computer Communications (INFOCOM'11), 2011.

[6] Mohammed Joda Usman, Abdul Samad, Ismail, Hassan Chizari, Ahmed Aliyu, "Energy-Efficient Virtual Machine Allocation Technique Using Interior Search Algorithm for Cloud Datacenter.", 2017 6th ICT International Student Project Conference (ICT-ISPC), 10.1109/ICT-ISPC.2017.8075327, 2017

[7] Christina Terese Joseph, Chandrasekaran K, Robin Cyriac, "A Novel Family Genetic Approach for Virtual Machine Allocation.", International Conference on Information and Communication Technologies (ICICT 2014), 2015

[8] Adrian, B., & Heryawan, L. (2015, November). Analysis of K-means algorithm for VM allocation in cloud computing. In 2015 International Conference on Data and Software Engineering (ICoDSE) (pp. 48-53). IEEE.