BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **18**, No 3
Sofia • 2018 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2018-0036

Pareto Based Virtual Machine Selection with Load Balancing in Cloud Data Centre

Ketaki Bhalchandra Naik[1,2], *G. Meera Gandhi*[1], *S. H. Patil*[3]

[1]*Sathyabama Institute of Science and Technology Chennai, India*
[2]*Bharati Vidyapeeth's College of Engineering for Women, Pune, India*
[3]*Bharati Vidyapeeth University College of Engineering, Pune, India*
E-mails: ketakin@gmail.com drmeeragandhii@gmail.com suhasharibhaupatil@gmail.com

***Abstract*:** *Cloud Data centers have adopted virtualization techniques for effective and efficient compilation of an application. The requirements of application from the execution perspective are fulfilled by scaling up and down the Virtual Machines* (*VMs*)*. The appropriate selection of VMs to handle the unpredictable peak workload without load imbalance is a critical challenge for a cloud data center. In this article, we propose Pareto based Greedy-Non dominated Sorting Genetic Algorithm-II* (*G-NSGA2*) *for agile selection of a virtual machine. Our strategy generates Pareto optimal solutions for fair distribution of cloud workloads among the set of virtual machines. True Pareto fronts generate approximate optimal trade off solution for multiple conflicting objectives rather than aggregating all objectives to obtain single trade off solution. The objectives of our study are to minimize the response time, operational cost and energy consumption of the virtual machine. The simulation results evaluate that our hybrid NSGA-II outperforms as compared to the standard NSGA-II Multiobjective optimization problem.*

***Keywords*:** *Cloud computing, data center, virtual machine, pareto optimal, NSGA-II, greedy.*

1. Introduction

Cloud users adopt the cloud computing services by utilizing the storage, computational and network resources available in the data centre. Cloud offers three important services in the form of Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [1]. The Data centres cater the demand of unlimited resources made by cloud users on pay per use basis with the help of virtualization. In virtualization [2], Virtual Machines (VMs) are created and allocated on top of the Physical Machines (PMs) in the form of computing instances of the PM. To speed up the overall operations of application, parallel execution of workloads is performed by VMs. In dynamic environment, the processing speed of virtual machines vary from one PM to another due to the unpredictable workload arrival and its allocation. This results in either overutilization or underutilization of VMs that

consequences load imbalance of data centre [3]. The scheduling of the workload on an appropriate virtual machine is an essential assignment to maintain stability and avoid poor system performance.

Therefore it is necessary to design an algorithm that could enhance the data centre performance by dynamically selecting the VM for workload execution in such way that all the VMS will perform relatively equal work at a particular point of time i.e. at intra data centre level. Hence load balancing and VM selection problem can be treated as Multiobjective and NP-hard problem [4].

1.1. Multiobjective optimization problem

The VM selection problem in a cloud environment needs to optimize multiple objectives. In general, these objectives conflict with each other. These contradictory objective functions, generate a set of optimal solutions. In the optimal solution set, no single solution is greater than other solutions as far as objective functions are concerned. These optimal solution sets are denoted as Pareto optimal solutions. A Multiobjective optimization problem with p decision variables and q objectives can be formulated as [5]:

(1) $$\text{Minimize } F(x) = \big(f_1(x), f_2(x), \dots, f_k(x)\big),$$

subject to:

$$g_i(x) \le 0 \quad \forall i \in [1, p],$$
$$h_j(x) = 0 \quad \forall j \in [1, q],$$

where $x \in \Omega$ and Ω is the decision space, R^n is the objective space, $F: \Omega \to R^n$ consists of k real valued objective functions $f_i(x)$ is the i-th objective function. $g_i(.)$ and $h_j(.)$ are the optional p number of inequality and q is number of equality constraints on the problem. These functions $f_1(x)$, $f_2(x)$, …, $f_k(x)$, are usually in conflict with each other. Similar to (1), in our article, we need to minimize the three conflicting objectives: response time, operational cost and energy consumption.

1.2. Pareto optimality

Pareto optimization is a technique for dealing with Multiobjective optimization problems [6]. This technique applies logical tactic concerning multiple contradictory objectives. Pareto optimization generates a large number of alternative optimal solutions as the best solution. However, with Multiobjective problems, the "best "is frequently reliant upon a user's inclinations often termed as the best compromise solution.

Generally, there is no solution that accomplishes the best solution for all objectives simultaneously. Therefore, the idea of Pareto optimality is suitable. A solution vector $x(1)$ is said to dominate the other solution vector $x(2)$, i.e., $x(1) \prec x(2)$, if both statement below are satisfied.

The solution vector $x^{(1)}$ is no worse than vector $x^{(2)}$ in all objectives, or $f_j(x^{(1)}) \le f_j(x^{(2)})$ for all $j \in \{1, 2, \dots, k\}$.

The solution vector $x^{(1)}$ is strictly better than $x^{(2)}$ if at least one objective, or $f_j(x^{(1)}) < f_j(x^{(2)})$ for at least one $j \in \{1, 2, \dots, k\}$.

The Pareto frontier is the plot of objective functions whose non dominated vectors are in Pareto set also called as feasible Pareto optimal points. Our algorithm

generates true Pareto fronts comprising of a set of optimal trade off solutions. VM selection with load balancing accomplished by calculating the load imbalance factor of the VM and allocating the workloads to VM as per its load capacity during its scheduling cycle. Our proposed Multiobjective approach provides a range of possible solution on convex Pareto optimal set more quickly and efficiently. In virtue of this, our article proposes a greedy based elitist Non-Dominated Sorting Genetic Algorithm-II called as G-NSGA2 strategy.

1.3. Contribution and Organization

The main contributions of this article are as follows:

1) A Multiobjective G-NSGA2 based strategy selects VM for workload scheduling and take care of data centre load balance.

2) To avoid trapping in local optima and to improve the convergence speed, our proposed algorithm uses greedy strategy along with the Multiobjective evolutionary algorithm NSGA-II.

3) Most of the approaches have given priority for the task scheduling rather than the selection of best computing resource, i.e., virtual machine on which the execution is to be carried out. Our algorithm selects the best VM first then schedules the workload. Therefore the success rate of faster execution of workload is better in our proposed algorithm as compared to other algorithms.

4) The proposed strategy generates Pareto optimal solution with the objectives of minimization in response time, operational cost and energy consumption.

The rest of the paper is organized as follows. Section 2 presents related work, Section 3 discusses the problem formulation while Section 4 explains the G-NSGA2 strategy implementation details. The strategy is evaluated through illustrations in Section 5. Section 6 concludes this research article.

2. Related work

Avoiding the load imbalance during static as well as dynamic scheduling of workload is the primary requirement of the data centre. When information related to workload and instances is known to the system prior to execution of an application, then it is called static scheduling or compile time execution. In case of dynamic scheduling the information related to workload and instances changes unpredictably then it is called run time execution [3]. Lots of paper are published on the load balancing technique and scheduling of workload in cloud environment. Some of the relevant papers are discussed here to understand the view of technique.

Qi Liu et al. [7] suggested an adaptive approach for allocating tasks evenly to improve time-space efficiency through map reduce function along with Multiobjective algorithm. Here the algorithm optimizes the execution time by map phase and prediction of execution time is achieved by reducer and the load balance is maintained by using Multiobjective algorithm. Instead Subhadra Bose Shaw [8] has proposed the algorithm for load balancer that will identify the least loaded VM available for scheduling of the tasks automatically on it to improve the response time of the resources. Here Author has used the static data to evaluate the

algorithm. A y y a p a z h a m and V e l a u t h a m [9] proposed neural network based load prediction technique in IaaS cloud environment. The algorithm uses genetically weight optimized neural network component for predicting the load of the host to create VM in future. The algorithm aims at reducing the response time and cost of the VM, while K. R a j e e v and T. P r a s h a r [10] discussed the bio inspired hybrid algorithm for efficient load balancing in cloud computing to minimize the average response time, processing cost and average data centre processing time. The combination of ant colony optimization and artificial bee colony algorithm have been used to handle the workload priority wise at multiple VMs. D e v i and U t h a r i a r a j [11] elaborated an improved weighted round robin algorithm for static and dynamic non-pre-emptive independent tasks scheduling to make cloud more efficient. F e n g et al. [12] proposed novel cloud load balancing mechanism for task scheduling and elastic cloud scaling to ensure user's Quality of Service (QoS) parameters with better load balance degree and completion time of the task. B h a t t and B h e d a [13] proposed flexible load balancing algorithm based on the grouping of the virtual machine as per the domain where the function will provide the details of overloaded machine within the same domain. L i u [14] proposed the load aware virtual machine placement algorithm for balancing the load within cloud data centre with the objective of reduction in network cost and energy consumption. G. S h i k h a, R. K. D w i v e d i and H. C h a u h a n [15] suggested dynamic synchronized throttled load balancing to avoid the under and over utilization of VM with the objective of maximizing response time and efficient allocation of coming requests. P h a m and N g u y e n [16] proposed ECRA-SA Algorithm for reducing the energy consumption and to improve the performance of data centre. The authors have considered the execution time and energy consumption as the objectives for virtual services provision in cloud computing. D. A t y a f and K. I. A r i f [17] discussed load balancing decision algorithm to manage and balance the load among virtual machines with the objective of minimization in completion and response time.

An important shortcoming of most current studies is that though they have considered multiple contradictory objectives for the optimization of the problem, still they are generating the single optimal solution by combining all the objectives using weight and constraint methods instead of Pareto optimal solution. Many studies have suggested the complete VM migration to balance the load and to overcome overutilization of VM that could lead to the poor response time of the system. Very few papers have suggested minimization in energy consumption along with load balance. As we are dealing with the utilization of resources from data centre, then carbon emission and energy consumptions should be considered as important parameters for the selection of VM for scheduling of workload.

3. Problem formulation

3.1. VM selector and load balancer model

The model of Multiobjective virtual machine selector and load balancer is shown in Fig. 1. Initially workloads of an application along with its requirements are uploaded by the cloud user to the workload portal. The workload analyser evaluates the

workload as per user's requirements, makes the grouping of workload and forwards it to centralized scheduler. The resource manager and selector component of the scheduler collects the capabilities, information of all VMs in the form of number of processing elements and processing capacity of VMs. This collected information is forwarded to the load balancer of the scheduler to compute the load of all VMs. The load balancer analyses the ratio between the number of workloads executing and the total number of VMs available. If the ratio is less than 1, then it conveys the scheduler to recognise a VM for the workload; else it will compute the load on each of the VMs based upon the data received from resource manager. The scheduler decides the threshold value dynamically as per the workload count. The VMs are classified as over utilized if the VM utilization is above the threshold value and if the utilization is below the threshold value then they are considered to be underutilized. Once the suitable VMs have been identified, the resource selector component of the scheduler runs the algorithm to fulfil the objectives of the cloud consumer. The algorithm generates the Pareto optimal list of VM for the scheduling of workload. Scheduler allocates the workload to the shortlisted VM for the execution of an application.

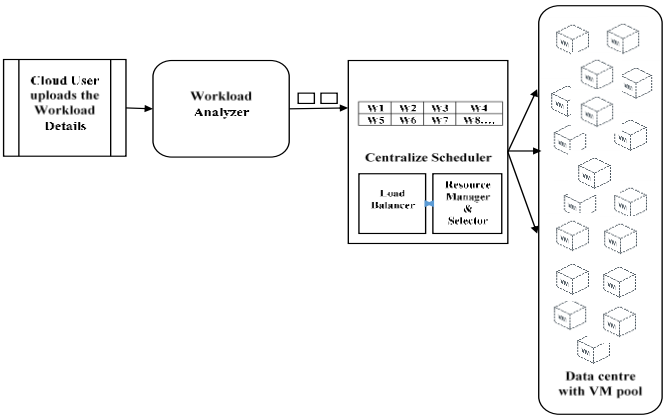

Fig. 1. VM selector and load balancer model

3.2. Virtual machine and workload model

Virtual machines are the computing instances of the data centre used for execution of independent/dependent and pre-emptive/non-pre-emptive workload of the application. In this article, we are considering non pre-emptive independent workload. The scheduler of the system dynamically selects the virtual machine as per its capacity and current load to allocate the workload. This maintains the load balance of the data centre otherwise load imbalance in data centre develops the additional challenges like reduction in response time, added operation and maintenance cost of resources

In general, virtual machine VM can be represented as VM={VM_1, VM_2, VM_3, …, VM_m}, as a set of M virtual machines that compute N workloads represented by the set $W = \{W_1, W_2, W_3, \ldots, W_N\}$. All the machines are heterogeneous and parallel in nature. Here the required data such as capacity of virtual machine in the form of

number of processing elements, memory size and network bandwidth required for VM are calculated in order to allocate the workload as per the applications' requirement. Cloud users upload the workload details in the form of workload length, expressed in Millions of Instructions (MI), as well as input output data file size of the workload. The workloads are in need of *p* amount of processing speed, *q* amount of memory size, *r* amount of storage space and *s* amount of bandwidth that will be fulfilled by selecting the VM of appropriate capacity for the execution of the workload.

3.3. Scheduling model

The virtual machine selection with the objectives of minimization in response time, reduction in operational cost and minimum energy consumption is performed using G-NSGA2 algorithm.

Response time. Response time is one of the Objective functions (Ob) for selection of virtual machine to balance the load. The scheduler is always in the search of VM having minimum response time for faster execution of workload. Hence response time can be stated as follows:

$$\text{Response Time} = \text{minimize } \text{Ob}_{j_1}(\text{WK}_i, \text{VM}_j), \tag{2}$$

where:

WK_i = independent workload *i* submitted for the execution of an application.

VM_j= virtual machine *j* selected for the scheduling of the workload.

Response Time of the VM for scheduling of the workload computed as

$$\text{Ob}_{j_1}(\text{WK}_i, \text{VM}_j) = \text{FT}_{ij} - \text{AT}_{ij} + \text{TT}_{ij}, \tag{3}$$

where,

FT_{ij} is Finish Time of workload *i* on VM*j*,

AT_{ij}=Arrival Time of workload *i* on VM*j*,

TT_{ij}=Transfer Time of workload *i* on VM*j*,

$$\text{TT}_{ij} = \frac{\text{DS}_i}{\text{VM}_j - \text{BW}} + \text{Delay}, \tag{4}$$

where:

DS_i is input data file size of the workload *i* to be transferred on VM_j,

VM_j is BW: Bandwidth of VM_j.

Operational Cost. To satisfy the user's requirement, centralized scheduler is always in search of VM that requires minimum Operation Cost (OCost) for the execution of workload *i* on virtual machine *j*. Hence it is represented and computed as

$$\text{OCost} = \text{minimize } \text{Ob}_{j_2}(\text{WK}_i, \text{VM}_j). \tag{5}$$

Operational Cost Calculation.

$$\text{Ob}_{j_2}(WK_i, \text{VM}_j) = \text{PRCost}_{ij} + \text{IOCost}_{ij}, \tag{6}$$

where:

PRCost_{ij} is Processing Cost of workload *i* on VM*j*,

IOCost_{ij} is Input Output Data Transfer cost for workload *i* on VM*j*,

$$\text{Ob}_{j_2} = \sum_{j=1}^{m} \sum_{i=1}^{n} \text{OCost}_{ij} \times X_{ij}, \tag{7}$$

$$\text{s.t. } \sum_{j=1}^{m} X_{ij} = 1,\ X_{ij} \in \{0,1\},$$

X_{ij} = 1 when virtual machine executes that workload *i*, otherwise 0,

$$\mathrm{PRCost}_{ij} = \mathrm{ET}_{ij} \times \mathrm{VM}_{j_{\mathrm{Cost}}}, \tag{8}$$

$$\mathrm{ET}_{ij} = \frac{\mathrm{WL}_i}{\mathrm{VM}_{j_{\mathrm{mips}}}}, \tag{9}$$

$$\mathrm{IOCost}_{ij} = \mathrm{TT}_{ij} + \mathrm{VM}_{j_{\mathrm{Cost}}}, \tag{10}$$

where:

ET_{ij} is Execution Time of workload *i* required on virtual machine *j*.

WL_{ij} is Workload Length of workload *i* computed in million instructions.

VM_{j_mips} is Computing Speed of Virtual Machine *j* in millions of instructions per 1 s.

Energy Consumption. To calculate the third objective, i.e., minimum energy consumption of virtual machines we focus on computational intensive application workload, therefore energy consumption caused by data storage and transfer are not considered. Minimization of Energy Consumption (EC) is represented as follows:

$$\mathrm{EC} = \text{minimize } \mathrm{ob}_{j_3}(\mathrm{WK}_i, \mathrm{VM}_j). \tag{11}$$

The total energy consumption is calculated as

$$\mathrm{ob}_{j_3} = \sum_{j=1}^{m} \sum_{i=1}^{n} e_{ij} \times x_{ij}, \tag{12}$$

where:

e_{ij} = energy consumption when workload *i* is executed on VM *j*.

x_{ij} = 1 when workload *i* consumes the energy of virtual machine *j*, otherwise 0.

Therefore, the VM selection in the cloud environment can be formulated as the following Multiobjective optimization problem.

$$\text{Response time} = \text{minimize } \mathrm{Ob}_{j_1}(\mathrm{WK}_i, \mathrm{VM}_j),$$

$$\text{Operational Cost} = \text{minimize } \mathrm{Ob}_{j_2}(\mathrm{WK}_i, \mathrm{VM}_j),$$

$$\text{Energy Consumption} = \text{minimize } \mathrm{Ob}_{j_3}(\mathrm{WK}_i, \mathrm{VM}_j).$$

Calculation of Load Imbalance factor. The sum of all virtual machines loads is defined as

$$\mathrm{LVM} = \sum_{i=1}^{k} \mathrm{VML}_i, \tag{13}$$

where *i*, represents the No of VMs in the data centre.

The load per unit capacity is defined as

$$\mathrm{LCVM} = \frac{\mathrm{LVM}}{\sum_{i=1}^{M} \mathrm{VMC}_i} \tag{14}$$

$$\text{Threshold } T_i = \mathrm{LCVM} \times \mathrm{VMC}_i, \tag{15}$$

where VMC_i = Capacity of the VM.

$$\text{If VM} \begin{cases} < T_i = \sum_{v=1}^{k} \mathrm{LVM}_v & \text{under loaded,} \\ > T_i = \sum_{v=1}^{k} \mathrm{LVM}_v & \text{overloaded,} \\ = T_i = \sum_{v=1}^{k} \mathrm{LVM}_v & \text{balanced.} \end{cases}$$

Before selecting the VM, the load balancer unit of scheduler calculates the loads of all VMs and then compares it with the threshold value that dynamically is set by the scheduler of the data centre. The VMs having less load value compared to the

threshold are considered as under loaded VMs. These VMs are used for workload scheduling on priority basis to avoid the load imbalance of the data centre.

4. G-NSGA2: NSGA-II based implementation details

In the process of VM selection, the centralize scheduler waits for specified duration, called waiting period or scheduling cycle. During this period, scheduler queues all the workloads and collects the capabilities of all VM from resource selector. The G-NSGA2 strategy is applied by the scheduler on the VMs to find the optimal mapping between the workload and VM. The result of the execution is a Pareto solution. Once the set of true Pareto fronts is generated, the scheduler selects first scheduling as per the cloud consumer's preference. The selected solution from the Pareto fronts is treated as a state for the virtual machine. This state will be an origin from which the next iteration of the strategy will make another execution on a collection of VM. The algorithm keeps iterating and selects the VM for mapping of workload until no more workload arrives.

4.1. G-NSGA2 algorithm steps

In year 2000, D e b et al. [18] implemented elitist Non-dominated Sorting Genetic Algorithm II (NSGA-II) that is a revision of the NSGA Multiobjective Evolutionary algorithm. NSGA-II offers greater transparency for finding the solution with the help of elitism operator, eliminating the parameter on the diversity operator and also reduces the complexity. Hence this article uses the NSGA-II algorithm to select the efficient virtual machine for workload scheduling in cloud data centres. Fast non dominated sorting and crowding distance are the major components of the NSGA-II algorithm. The procedure involved is as follows.

To initialize with, parent population PI of size S is generated using random and greedy method. Details related to population initialization are present in Section 4.3. From PI an offspring QI of size S is constructed by adopting tournament selection, crossover and mutation operators. PI and QI are aggregated to generate total initial population RI of size $2S$. Hereafter the algorithm keeps iterating to improve the fitness of individual in the population. The new population PI+1 of size S is obtained from population RI of size $2S$ using non-dominated sorting using and calculating crowding distance. The crowding distance of each individual is calculated as per the Euclidian distance between each individual arranged in front, based on their fitness function, comparison and sorting of the results. The individuals in the boundary are always selected because they have infinite distance assignment. The Crowding Distance (CD) of other solutions are

$$\mathrm{CD}_J = \sum_{I=1}^{K} \frac{|V_{I(J+1)} - V_{I(J-1)}|}{V_I \max - V_I \min}, \tag{16}$$

here, K is the number of objective functions and CD is the crowding distance of the J-th individual. V_{IJ} is the value of the J-th individual in the I-th objective function. The final solution is the one which has the largest crowding distance by means of NSGA II algorithm. Finally QI+1 offspring is obtained by using genetic operators and crowding distance on PI+1 population. At the end RI+1 of size $2S$ is obtained by

combining PI+1 and QI+1. The solutions obtained through NSGA-II are said to be Pareto optimal if they are non-dominated by any other solution in the solution space. RI+1 of size 2*S* is called as Pareto optimal set if the real diversity and capabilities are present in the individual of the set.

4.2. Problem encoding

Fig. 2 depicts the encoding mechanism where individual solution as a vector of integers. In our model, the index of the cell represents the workload id that are scheduled, the integer in each cell of vector describes the VM to which the workload is assigned. Here each workload's request is allocated to one VM but VM is able to handle more than one workload. This encoding notifies about the number of workloads available for scheduling during the first scheduling cycle.

W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
1	2	5	4	2	5	3	4

Fig. 2. Example of chromosome in encoding solution

4.3. Population initialization

Population initialization is an important steps in NSGA2. In our strategy, the initial population is generated in combination of two methods. The first method aims to select the VM with minimum response time using greedy algorithm while second method adds diversity to the population by using random method. The VMs having high speed and good processing capacities are identified by calculating response time using (3). Then sorting of all VMs as per response time is performed. After that first VM is selected and the VMs having the response time more than first are removed from the solution and thus recursively the action is performed on all the VMs of the set. The solution set of VM having minimum response time is generated before the start of next scheduling cycle.

4.4. Fitness function

The objective functions, minimization in operational cost (5) and reduction in energy consumption (11) are used as fitness function to evaluate the population generated by the greedy and the random method. The fitness is evaluated for each iteration until the termination criteria does not meet.

4.5. Genetic operators

Selection. An offspring is generated from the parent solution. The crowding tournament technique is used for selecting the parent solution from population. The selection strategy consists of elitism, i.e., selecting best one from randomly choosing two solutions of the population by replacing the old worst individual from solution. The solutions belonging to smaller fronts are always good. The same front solutions are compared on the basis of crowding distance. The solution having bigger crowding distance wins the tournament.

Crossover. Crossover operator generates better children by combining two parent solutions. Many crossover operators have been proposed in the literature [19]. In this paper, we use Two-Point crossover (TPX) [19] which showed a good performance to solve the problem.

Mutation. Mutation operator has low complexity and provides solution diversity. These properties of mutation boosts the capabilities of NSGA-II to find the real Pareto front. Here, we use swap mutation technique [20]. In a chromosome, two genes are randomly selected and the value of their alleles is exchanged. Before starting the next scheduling cycle the state of the VM is stored by calculating the load imbalance factor using (15) of each VM. The VMs having less threshold, are selected for workload allocation of next cycle.

Thus to solve the scheduling problem with best possible solution in an acceptable time frame, a combination of exploitation and exploration from two population based heuristic algorithms are used. NSGA-II has the ability of expanding the search space in the form of exploration whereas greedy method has the ability of finding the optima around the good solution by exploitation and thus helps to avoid the algorithm being trapped in local optima. This hybrid algorithm G-NSGA2 is designed to achieve minimum response time and reduction in operational cost with minimum energy consumption. The flow of G-NSGA2 process is shown in Fig. 3.

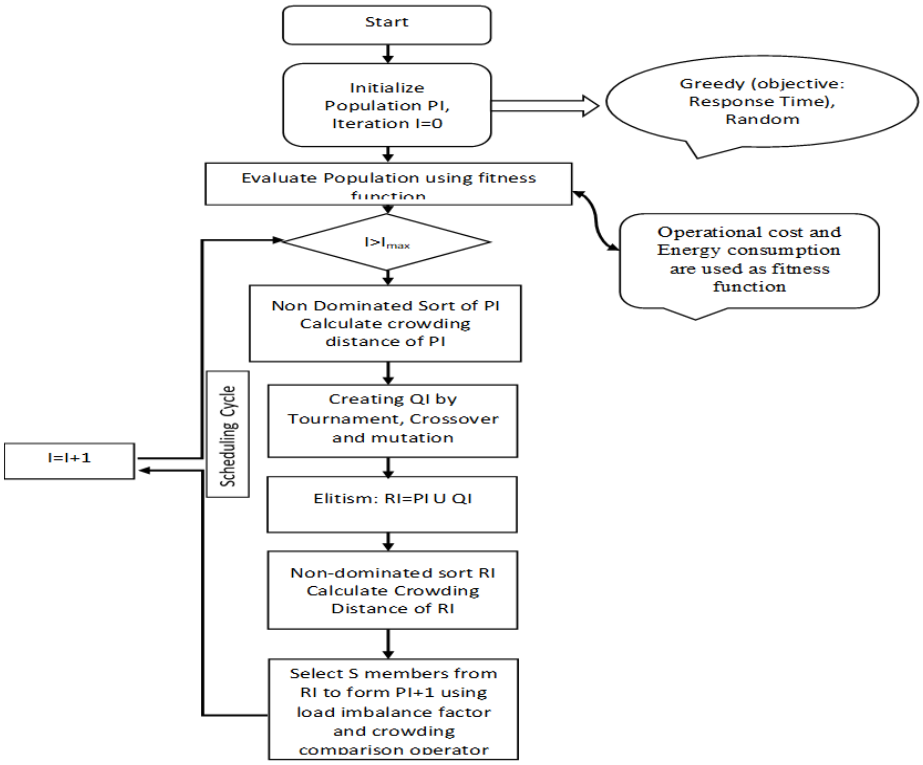

Fig. 3. Flowchart of G-NSGA2 procedure

5. Performance evaluation

This section explains the simulation results of the proposed algorithm and its comparison with the state of the art NSGA-II Multiobjective algorithm. The simulations were carried out using Python as an implementation language on an Intel(R) Core(TM) i7-5500U with 2.2 GHz processor and 16 GB RAM with Linux platform. The specifications and parameters used for demonstration of our proposed algorithm in cloud environment, are given in Table 1.

Table1. Experimental setting

Parameter	Value
Network bandwidth	1 MB per 1 s
Boot time and shutdown time of VM	0.5 s
Performance variance of VM	24%
Client execution price	$0.70/CPU per 1 h
Scheduling cycle	80 s

5.1. Performance metrics

Here we have used the normalized fitness function and response time value by applying the next normalization formula on (3), (7) and (12):

(17) $$\widehat{x} = \frac{x_i}{\max\limits_{j=1 \text{ to } n} x_j},$$

where $\widehat{x}$ = Normalized value of x_i.

We have used max normalization to normalize the absolute fitness value for easy comparison and visualization of overall quality of the result. The maximum value is mapped to one and the rest of the values lie in the interval [0, 1].

5.2. Dataset

We have used workloads traces from Feitelson's Parallel Workload Archive (PWA) [21].This helps to simulate a heavy workload scenario as workloads play an essential roles in process of scheduling. The traces are shortlisted based on the high utilization rate and the offered load on the cloud during the specified period.

5.3. Result analysis and performance evolution

We compare the average makespan and average response time of G-NSGA2 with NSGA-II [22]. Multiobjective optimization algorithms comparison as per the objectives are shown in Figs 4, 5 and 6, respectively. All the values of response time, energy consumption and operational cost are normalized to visualize the effect of true Pareto fronts. The NSGA-II and G-NSGA2 algorithms are initialized based on stochastic solutions into consideration as both have better search ability. The Pareto fronts acquired by NSGA-II are dominated by the G-NSGA2. The true Pareto fronts for Response time vs. energy consumption are shown in Fig. 4.

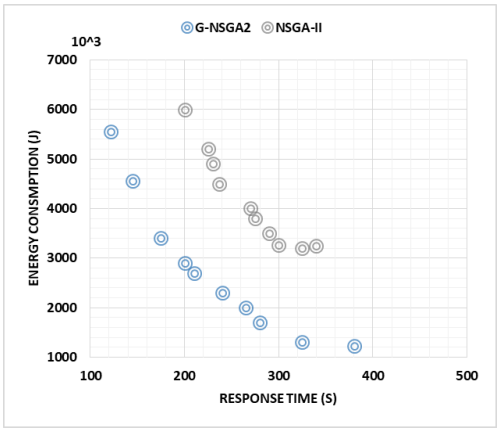

Fig. 4. Pareto curve for response time vs. energy consmption

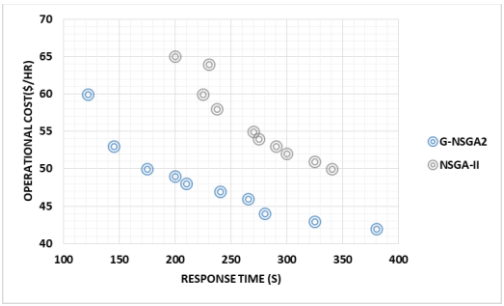

Fig. 5. Pareto curve for response time vs. operational cost

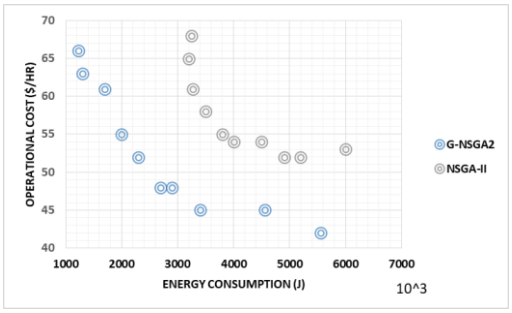

Fig. 6. Pareto curve for energy consumption vs. operational cost

We can see that the results of G-NSGA2 are better than NSGA-II because of better evolutionary strategy mixed with good a combination of cooperative and competition mechanisms. The diversity is more in G-NSGA2 as compared to NSGA-II. This is because the G-NSGA2 achieves more balanced load distribution between PMs and selects the VM having less degree of imbalance. In Fig. 5 we can see that Pareto fronts of Response time vs. operational cost of G-NSGA2 are better than that of NSGA-II. Since the G-NSGA2 uses the greedy method for finding objective of response time along with fitness values, this avoids fronts to fall in local optima.

Fig. 6 represents the Pareto fronts for Energy consumption vs. Operational cost. Here also G-NSGA2 has shown significant improvement and average gain because of elasticity in cost value of G-NSGA2 and good combination of evolutionary algorithm with the local search heuristic to achieve the optimal result.

Table 2. Parameter Setting

Specifications	Values
Population Size	500
Maximum Iterations	200
Crossover Probability (Pc)	0.7
Mutation Probability (Pm)	0.6
Crossover Type	Two point Crossover
Mutation Type	Swap Mutation
Tournament group size	2

6. Conclusion

In this article, we presented the problem of VM selection and scheduling of workload with the objectives of minimization in response time, operational cost and energy consumption. We proposed a novel multi-objective G-NSGA2 strategy, capable of computing trade off solutions between response time, operational cost and energy consumption. This strategy trusts on empirical models for predicting the energy consumption and operational cost of workload. We compared G-NSGA2 with NSGA-II, the contemporary Multiobjective algorithm for optimising VM response time, energy consumption and operational cost. The experiments presented that G-NSGA2 calculated Pareto sets of higher quality than NSGA-II. Finally, we analysed the G-NSGA2's response to peak workloads and for matching trade-offs in objectives.

***Acknowledgements*:** This work has been partially supported by the Department of Science and Technology, India and School of Computing, Sathyabama Institute of Science and Technology, Chennai for providing the facilities to do the research under the DST-FIST Grant Project No SR/FST/ETI-364/2014

References

1. B u y y a, R. K., et al. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. – Future Generation Computer Systems, Vol. **25**, 2009, No 6, pp. 599-616.
2. L i, W., J. T o r d s s o n, E. E l m r o t h. Modelling for Dynamic Cloud Scheduling via Migration of Virtual Machines. – In: Proc. of 3rd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom'2011, 2011, pp. 163-171.
3. D a v i s, L. J., L. J. D a v i s. Selection of Load Balancing Parameters. – Vol. **9655**, 2015, No October.
4. U l l m a n, J. D. NP-Complete Scheduling Problems. – Journal of Computer and System Sciences, Vol. **10**, 1975, No 3, pp. 384-393.

5. S r i n i v a s, N., K. D e b. Multiobjective Optimization Using No Dominated Sorting in Genetic Algorithms. – Evol. Comput., Vol. **2**, 1995, No 3, pp. 221-248.
6. Y a i r, C. Pareto Optimality in Multiobjective Problems. – Applied Mathematics and Optimization, Vol. **4**, 1977, No 1, pp. 41-59.
7. L i u, Q., et al. An Adaptive Approach to Better Load Balancing in a Consumer-Centric Cloud Environment. – IEEE Transactions on Consumer Electronics, Vol. **62**, 2016, No 3, pp. 243-250.
8. S h a w, S. B. Balancing Load of Cloud Data Centre Using Efficient Task Scheduling Algorithm. – International Journal of Computer Applications, Vol. **159**, 2017, No 5, pp. 1-5.
9. A y y a p a z h a m, R., K. V e l a u t h a m. Proficient Decision Making on Virtual Machine Creation in IaaS Cloud Environment. – Vol. **14**, 2017, No 3, pp. 314-323.
10. R a j e e v, K., T. P r a s h a r. A Bio-Inspired Hybrid Algorithm for Effective Load Balancing in Cloud Computing. – International Journal of Cloud Computing, Vol. **5**, 2016, No 3, pp. 218-246.
11. D e v i, D. C., V. R. U t h a r i a r a j. Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks. – The Scientific World Journal, Vol. **2016**, 2016.
12. F e n g, Y., et al. A Novel Cloud Load Balancing Mechanism in Premise of Ensuring QoS. – Intelligent Automation & Soft Computing, Vol. **19**, 2013, No 2, pp. 151-163.
13. B h a t t, H., H. A. B h e d a. Enhance Load Balancing Using Flexible Load Sharing in Cloud Computing. – 2015, No September, pp. 4-5.
14. L i u, C. A Load Balancing Aware Virtual Machine Live Migration Algorithm. – In: Proc. of International Conference on Sensors, Measurement and Intelligent Materials, 2016.
15. S h i k h a, G., R. K. D w i v e d i, H. C h a u h a n. Efficient Utilization of Virtual Machines in Cloud Computing Using Synchronized Throttled Load Balancing. – Proc. of 1st International Conference on Next Generation Computing Technologies (NGCT'15), IEEE, 2015.
16. P h a m, N. M. N., H. H. C. N g u y e n. Energy Efficient Resource Allocation for Virtual Services Based on Heterogeneous Shared Hosting Platforms in Cloud Computing. – Cybernetic and Information Technologies, Vol. **17**, 2017, No 3, pp. 47-58.
17. A t y a f, D., K. I. A r i f. An Efficient Load Balancing Scheme for Cloud Computing. – Indian Journal of Science and Technology, Vol. **10**, 2017, No 11.
18. D e b, K., S. A g r a w a l, A. P r a t a p, T. M e y a r i v a n. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. – Parallel Probl. Solving from Nat. PPSN VI, 2000, pp. 849-858.
19. R u i z, R., C. M a r o t o, J. A l c a r a z. Two New Robust Genetic Algorithms for the Flow Shop Scheduling Problem. – Omega, Vol. **34**, 2006, No 5, pp. 461-476.
20. M i t s u o, G., F. A l t i p a r m a k, L. L i n. A Genetic Algorithm for Two-Stage Transportation Problem Using Priority-Based Encoding. – OR Spectrum, Vol. **28**, 2006, No 3, pp. 337-354.
21. F e i t e l s o n, D. G. Parallel Workload Archive, 2007. **http://www. cs. huji. ac. il/labs/parallel/workload**
22. K a l y a n m o y, D., et al. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. – In: Proc. of International Conference on Parallel Problem Solving From Nature. Springer, Berlin, Heidelberg, 2000.

Received 06.07.2018; Second Version 21.08.2018; Accepted 27.08.2018