Multiobjective Workload Scheduling in Distributed Datacenters

Ketaki Naik
Dept. of Computer Science & Engg.
Sathyabama Institute of Sciecne & Technology ,
Chennai,India-600119
&
Dept. of Information Technology
Bharati Vidyapeeth's College of Engg. for Women,
Pune,India-411043
ketakin@gmail.com

G.Meera Gandhi
Dept. of Computer Science & Engg.
Sathyabama Institute of Sciecne & Technology ,
Chennai,India-600119
drmeeragandhii@gmail.com

S.H.Patil
Dept. of Computer Engg
Bharati Vidyapeeth University college of Engg.
Pune,India-411043
suhasharibhaupatil@gmail.com

***Abstract*—Cloud computing is now a popular field for indulging infrastructure–as-a-service (IaaS). IaaS defers, on demand virtualized computing resources through datacenters. Cloud users acquire computing infrastructure for the execution of application workloads. However due to increasing data volumes and parallel processing of workloads on limited resources of single data center, workloads are shifted to the other datacenters over the peak time. Therefore scheduling of workloads in distributed datacenter is a challenge as each workload executes across disbursedenvironment having heterogeneous resources and also requires coordination among the datacenters. In this paper, we propose MO-GRASP (Multiobjective Greedy Randomized Adaptive Search Procedure) based heuristic to solve the multi-objective workload scheduling problem. The model objectives are minimization in workload completion time i.e. datacentermakespan and energy consumptionwhilemaximizing the average datacenter utilization. The simulation results show that the proposed algorithm outperforms as compared to other Multiobjective algorithm.**

Keywords—cloud computing, Multiobjective, GRASP, workload scheduling, datacenter

I. Introduction

Cloud computing enables on demand resources to be offered as infrastructure-as-a-service (IaaS), platform-as-a service (PaaS) and software-as-a-service (SaaS) for the transaction and execution of online application workloads [1]. Application workloads,produced from online transactions,require parallel processing of the data to generate speedy responses for the cloud users. Therefore the compute,storage and networking resources of the datacenter are virtualized to accomplish the amount of work within certain period of time [2].Online applications of web and database servers such as user behaviour analysis,web search, and business intelligence generate significant workloads for the datacenter.Each application's workload composed of a numerous trivially parallel tasks. Application completes its execution after the completion of all its workload. To contribute in growing demand of several data analytics applications, and to minimize the slower responses from web server due to overloaded web traffic, Amazon [3], Microsoft [4] and Google [5], as such many cloud providers schedule workloads in multiple distributed datacenters.

Traditional methods perform centralized application execution, with each application executing within single datacenter. In such case, as and when an application requires data from several datacenters, a usual method is, to gather primarily all the needed data from various datacenters at a centralized location. However such an approach consequences in considerable network traffic as bulks of data endure to raise unexceptionally and increased workload completion time. Moreover it is gradually unfeasible to get unlimited resources on single datacenter and replication of large data set through many datacenters[6]. Thus, a current trend is executing a workload's tasks at respective datacentres where the required data warehoused and then accumulating the results at workload completion time rather than collecting the data at single datacenter, i.e. distributed execution of workloads. Though encouraging, the distributed execution of workload poses new challenges for the workload scheduling. Completing a part of the workload rapidly at one datacenter does not generate quicker overall workload completion time due to the dynamic and heterogeneous computing resources of the datacenter. In addition, workload tasksprocessed at a specific datacentre depends upon the availability of the data in the datacentre, thereby increases the complexity of the problem. Hence, it is essential to schedule the workload correctly to balance computation and communication overheads.

To address the challenges outlined above, this paper marks the following contributions.

(1)Design and development of two-level schedulers' module for the partition and execution of workload tasks in distributed environment of datacentres. (2) Design and implementation of Multiobjective GRASP based heuristic for the scheduling of workloads with the objectives such as minimization in workload completion time, maximization in average datacentre utilization and minimization in energy consumption.(3) Simulation and Demonstrationof multiple conflicting criteria that opt for nearby optimal schedule.

Organization of the paper is as follows: Section 2 describes related work. Section 3 depicts models and problem formulation. In addition, section 4 explains MO-GRASP based procedure and section 5 provides experimental evolution followed by conclusion.

II. Related Work

In cloud computing, with the constant infrastructural and middleware development, scheduling in distributed datacenter is one of the topic that has gained the attention of researchers and scientists.Several task-scheduling algorithms implemented for heterogeneousdistributed computing infrastructures such as cluster, grid [7,8] and some of these are also extended for cloud computing.

L. F. Bittencourt et al. [9] described taxonomy of scheduling in distributed system from the perspective of cloud computingthat encompasses the scheduler organization and the development of scheduling algorithms in the systems. While X. Tao et al. [10]has suggested profit aware task scheduling for improving the application performance within reasonable time and a method for absent deadline to maximize the profit inside the datacenter network.In addition to this,M. Moca et al. in [11] investigated fault tolerant and trust aware Promethee scheduler for scheduling of Bag-of-tasks application on distributed computing infrastructure like grids and clouds. The centralized scheduler is designedto maximize the user satisfaction in the form of price, expected completion time and trust. Also,S. Iturriaga, and A. Tchernykh[12] presented Multiobjective approach for workflow scheduling in distributed datacenters with the objectives of makespan, energy consumption and deadline violations. Two level schema is designedto schedule the jobs in higher level and lower level clusters.M. Guzek et al. [13] explored the problem of workflow scheduling application using MOcell, NSGA-II and IBEA algorithms on heterogeneous, DVFS enabled distributed platform with the objectives of schedule length and minimization of energy consumption.R. K. Jena[14] implemented task scheduling nested PSO algorithm to optimize the energy and processing time of the tasks. However, many scheduling approaches mentioned above haveneither considered the superiority of the result nor assessed feedback that tends to local optima. This paper considers the superiority of the result and performance feedback to adjust itself rendering to feedback grades. In addition, we used parallel and distributive processing of workload to reduce the workload execution time. Until now, there is little research done on the criteria considering the objectives like workload completion time i.e. datacenter makespan, averagedatacenter utilization and energy consumption. To the best of our knowledge, the solution methodology proposed does not overlap with existing techniques for scheduling problems.

III. PROBLEM MODELING

The problem we consider in this work is the scheduling of workload application on distributed datacenters. We target the minimization of workload completion time (datacenter makespan) and energy consumption along with maximization of average-datacenter utilization.
In this model as shown in Fig. 1 we have used two-level workload-schedulers: (1) The global meta-scheduler and (2) local scheduler of the datacenters. The global meta-scheduler consists of four main components as Workload Monitor and Analyser, Datacentre Discovery, Energy Calculator and Dispatcher. Workload Monitor & Analyser monitors and analyses the requirements of resources for the parallel execution of workload. Datacentre Discovery helps in identification of datacentre as per the requirement of resources for workload. Further, Energy Calculatorsuggests the datacentre that consumes minimum energy for the processing of workload.The workload execution takes place only when the actual energy consumption (DC_{Energy}) is less than the threshold energy value ($Energy_{Threshold}$). Finally, Dispatcher schedules the workload's task to the local scheduler of the datacentre as per the requirement of workload.

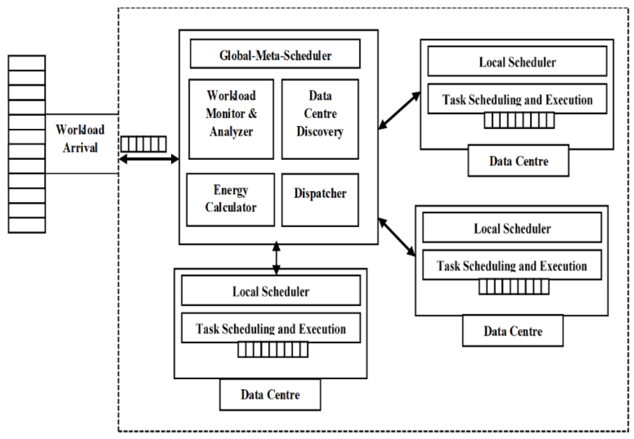

Fig. 1. Model of workload scheduling in distributed datacenters

The global meta-scheduler acts as an interface for group of datacenters *DC* spreadacross geographical areas and schedules the workloadto the local schedulers of the datacenters by partitioning it into several tasks. The datacenter model used in this work consists of a set of H datacenters as shown in equation (1).

$$DC = \{DC_1, DC_2, DC_3, \ldots\ldots, DC_H\} \qquad (1)$$

Datacenters provide on demand resources and computing facilities over the Internet in the form of VMs (Virtual Machine). The VMs deployed in datacenters are of different types and characteristics. As such, they may have different number of CPUs of different cycle times, memory capacity and network bandwidth. Each virtualized datacenter comprises set of heterogeneous VMs, which have different performance and represented as shown in equation (2).

$$VM_i = \{vm_{i1}, vm_{i2}, vm_{i3}, \ldots, vm_{im}\} \qquad (2)$$

Where, $1 \leq i \leq H$. The performance of vm_{ij} represented as $Speed_{ij}$. Therefore, we can say that the speed of datacenter$DC_k$is as follows:

$$DCspeed_k = \sum_{i=1}^{m} Speed_{ij} \qquad (3)$$

Where, $1 \leq k \leq m$. Every datacenter has local scheduler that acts as an entry point and knows the status of the VMs of its datacenter. To reduce the fragmentation of VM allotment and to maximize the datacenter utilization, the local scheduler uses easy backfilling technique for tasks allocation within datacenter. Backfilling technique is the extended version of the first come first served scheduling technique. Generally, the workload sequence decided by the global meta-scheduler or the local scheduler itself. The local scheduler executes the tasks from its queue in next computing slot at datacenter site. In addition, all datacenters inform their developments to the global scheduler, in provision of effective global workload scheduling verdicts.Therefore,the workload-level scheduling verdictsare made insynchronization of the global and local schedulers.

Execution of application workload takes place within an individual datacenter and does not have pre-emptive priority.In addition, the workload completes its execution only after all of its tasks are completed; consequently, the workload completion time computed after the execution of end task.

Consider a set of n workloads available in each application received at the global meta-scheduler. Each workload is composed of trivially parallel and independent task. equation (5) shows the representation of workload W_p

$$A = \{W_1, W_2, W_3, \dots\dots, W_n\} \quad (4)$$

$$W_p = \{t_{p1}, t_{p2}, t_{p3}, \dots\dots, t_{pb}\} \quad (5)$$

Assumption, for any workload $W_p \& W_l (p \neq l)$, i.e. total number of tasks in workload $W_p$are not equal to the total number of tasks present in workload W_l. Each task $t_{pc}$of workloadW_p has its corresponding task length represented, as TL_{pc}.$FS_{pc}$is the file data size of the task $t_{pc,}$ required for the I/O data processing of the task.

A. Scheduling Model

- Workload completion time (datacentermakespan):

Here we calculate the datacenter makespan (Ms) as,the total time required to complete all the workload tasks by the available datacenter.

Let us consider,

$Ms(DC_k)$	Makespan of datacentre k
where,$1 \leq k \leq H$,	
$\lvert W_p \rvert$	total number of tasks of the workload W_p
Where, $1 \leq p \leq n$	
AT(pc)	start time of the task of workload p
CT(k)=0	completion time of the last task on data centre k
F(pc,k)	Boolean variable defined as follows

$$f(pc,k) = \begin{cases} 1, & tpc \text{ is assigned to } DCk \\ 0, & otherwise \end{cases} \quad (6)$$

Therefore, makespan of individual datacenter, DC_kmathematically expressed as follows:

$$Ms(DC_k) = \sum_{p=1}^{n} \sum_{c=1}^{|W_p|} EET(pc,k) \times f(pc,k) + AT(pc) - CT(k) \times f(pc,k)$$
(7)

Expected Execution time of workload task obtained by aggregating execution time of task with transfer time of the file data and represented as shown in Equation (8).

$$EET(pc,k) = \frac{TL_{pc}}{DCspeed_k} + DTT(pc,k) \quad (8)$$

Where, *DTT (pc, k)* - filedata transfer time of the task. It is the ratio of filedata size required for the task execution to the bandwidth of the datacenter

$$DTT(pc,k) = \frac{FS_{pc}}{BW_k} \quad (9)$$

Therefore, the overall makespan of the datacenter using equation (7) represented in equation (10).

$$O1(V) = Ms = \max(Ms(DC_k), 1 \leq k \leq H \quad (10)$$

For getting the optimal result of the algorithm, overall makespan should be least.

- Average Datacenter Utilization (AU)

At any given time, for a Datacenter$DC_{k,,}$the utilization $U_k$is shown in equation (11). Where n is the entire number of workloads running at that time. $U_k.|W_p|$ is the datacenter usage of the workload $|W_p|$. The average datacenter utilization (AU*)* is the average utilization of all the datacenters represented in equation (12).

$$U_k = \sum_{p=1}^{n} U_k.|W_p| \quad (11)$$

$$O2(V) = AU = \frac{1}{H} \sum_{k=1}^{H} U_K \quad (12)$$

- Energy Consumption (EC)

As power consumption of the datacenter affected by the utilization of the resources within datacenter, we adopted the energy model stating the linear relationship between datacentre utilization and energy consumption [15]. The actual energy consumption EC_k at datacenterDC_k at any given time is as shown in equation (13).

$$O3(V) = EC_k = (P_{max} - P_{min}) \times U_k + P_{min} \quad (13)$$

Where, P_{max} is the maximum power consumption i.e. 100 % utilization aka peak load consumption. P_{min} is the minimum power consumption in the active mode.Therefore, workload scheduling in distributed datacentre is formulated as Multiobjective problem as stated below.

$$\left\{ \begin{array}{ll} \text{Workload Completion Time:} & \text{Minimize } O1(v) \\ \text{Average Datacentre Utilization:} & \text{Maximize } O2(V) \\ \text{Energy Consumption:} & \text{Minimize } O3(V) \end{array} \right\}$$

The weighted sum is the most common approach for coping with Multiobjective optimization problem where all objectives are aggregated to a weighted combination as shown in equation (14).

$$F(c) = \omega \times O1(V) + (1-\omega) \times O2(V) + (\omega) \times O3(V) \quad (14)$$

Where, $\omega \in (0,1)$ to determine the optimal point per optimization run.

IV. Proposed multiobjective GRASP procedure

GRASP is an iterative randomized metaheuristic technique effectively used in routing and scheduling problems. [16] Proposed the guidelines for solving combinatorial optimization problems based on GRASP.The benefit of GRASP associated with other heuristics is that the size of the candidate list and the number of grasp iterations these two parameters are required to tune. GRASP seems to be economical with respect to the superiority of the gained solutions and is easier to implement. It comprises of two phases: construction phase and local search.
In this proposed MO-GRASP algorithm, we have adapted the finite weight vectors to generate non dominated front solution of multi-objective search with diversity and convergence. More insights on Multiobjective GRASP has been provided in [17] for path relinking problem.

Algorithm 1 : The proposedMO-GRASP algorithm for workload scheduling

```
1.  Do Until maximum iterations are not completed
2.  WLschedule=Generateschedule(workload)
3.  IfWLschedule dominates the bestWLschedule
4.  Set bestWLschedule=WLschedule
5.  End if
6.  End do
7.  Function: Generateschedule(workload)
8.  soln  = MO-construct-soln(workload)
9.  NHsoln = MO-local-search(soln)
10. IfNHsoln is superior than soln
11. Print NHsoln
12. Else
13. Printsoln
14. End if
15. End functionGenerateschedule
16. Function: MO-construct-soln(workload)
    // construct the new front at each generation
17. Do Until schedule is not completed
18. Set W = all unassigned tasks of workload
19. Generate RCL foreach t of workload W
20. {s}=sub-soln
21. {s}=select datacentre randomly
    // generation of  new-front
22. foreach task t of workload W from its RCL
23. soln =soln  U {s}
    // generate the best front i.e. non dominated set
24. Revise information  of RCL for non-dominated
    solution
25. End do
26. Printsoln
27. End functionMO-construct-soln
28. Function: MO- local-search (soln)
29. Set NHsoln = optimum local solution
    //if NHsoln is not dominated by any other member
    of front
30. Print NHsolution
    //return front and remove dominated solution from
    front
31. End function MO- local-search
```

Algorithm 1, explains the MO-GRASP to obtain the optimal solution by conducting number of iterations for a workload-scheduling problem. At each iterative step, an algorithm generates solution and the best non dominated solution considered, as a final schedule indicated by line no 2 to 6. The algorithm terminates after a completion of a certain number of iterations by generating feasible solution. In general, each iteration performs construction and local search phases to generate feasible solution. The construction phase for workload scheduling is indicated at line no 8 and line no 16 to 27 in Algorithm 1. The prerequisite for the workload-scheduling problem to generate feasible solution is that a workload finishes its execution only after all its tasks complete their execution.

Algorithm 2 : Construction phase procedure

```
1.  Function: MO-construct-soln
2.  Do Until schedule is not completed
3.  Set feasible-tasks = unassigned ready tasks
4.  Set {s} = schedule(feasible-tasks)
5.  soln = soln U{s}
6.  End do
7.  Printsoln
8.  End function construct-soln
9.  Function: schedule(wtasks)
10. Set feasible-tasks=wtasks
11. match
12. Do Until all workload tasks are not scheduled
13. foreach  t = feasible-tasks do
14. feasible-datacentres = available datacentres for t
15. foreach datacentre DC do
16. Compute Objective Function Value (OVF)
    increase makespan (t, DC)
17. match = match U (t, DC)
18. end for
19. end for
20. min I = minimum OVF increase over feasible
    match
21. max I= maximum OVF increase over feasible
    match
22. set $\alpha$ ( discrete greedy value) as an energy
    consumption threshold calculated by energy
    calculator
23. select feasible- match as RCL whose makespan
    increase <= min I + $\alpha$ (max I – min I)
24. select (t' DC')  as a  random candidate  map from
    RCL –match
25. extract t' from feasible-tasks
26. soln = soln U (t', DC')
27. End do
28. Printsoln
29. loop the steps 2 to 26 until all tasks are assigned
30. Compute OFV for complete workload schedule
31. loop all the steps for n-times for changed values of
    $\alpha$
32. Compute maximum datacentre utilization
    corresponding to minimum makespan
33  End schedule
```

In construction phase, a restricted candidate list archives the best datacentres as candidates for workload scheduling considering the minimum energy consumption as compared to energy threshold set by the global scheduler. Here, datacentres are selected randomly for the allocation of each workload task using value based RCL as shown in Algorithm 1 line no 19. Value based RCL consists of solution candidates whose performance-evaluated values are better than given threshold. After assigning datacentre to a task, the local scheduler updates datacentre information to the global scheduler. Then global scheduler endures to work on other unassigned tasks. Algorithm 2 illustrates the complete execution of the construction phase for workload scheduling. The global scheduler computes the OVF (makespan) increase for each unassigned task on each datacentre available to execute the task. The scheduler randomly chooses a task mapping from the task and datacentre match whose makespan increase is less than $minI+\alpha(maxI-minI)$, where α is a bound to determine how much deviation is permissible for generating RCL for each task. Here, we set the threshold energy value to α *as discrete greedy value* to generate the RCL of datacentre having a minimum energy consumption.

After the construction of feasible solution. Local search performs the improvement on the solution. The local search procedure spawns a novel solution by exploring local optima in the neighbourhood of the existing solution. During the construction phase, if the performance of the novel solution in the form of makespan and datacentre utilization are better than the current solution then it will substitute the existing constructed solution.

V. PERFORMANCE EVOLUTION

The simulation results of proposed MO-GRASP algorithm obtained using cloud simulator CloudSim 3.0.2 with NetBeans IDE 8.0 installed on Windows 10 using Intel core i7 4.90GHz processor.The experimentation randomly generates the datacentres with different capabilities in termsof RAM, number of processors and their processing speed,and bandwidth. Each datacentre is also capable of creating virtual machine instances. The workloads produced via Feitelson's parallel workload archive[18]. Around 200-1000, workload requests generated randomly in the simulation. The simulation results that comprise of minimization of overall completion time of workload i.e. (DatacenterMakespan), minimization of energy consumption and maximization of average datacentre utilization compared. The derived objective function values allocate identical weights to completion time and energy consumption throughout the optimizationprocedure. There is a trade-off between maximization and minimization of objective values and ourproposed algorithm solves the trade-off issue by allocating suitable weightage to those three aspects. The comparison of the proposed MO-GRASP made with popular metaheuristic genetic algorithm MOGA (Multiobjective Genetic Algorithm).MOGA [19] represents solutions in the form of chromosomes and inherits the genetic algorithm principle survivalof the fittest. The parameters such as selection, crossover, andmutation play an important role in convergence of local optima.

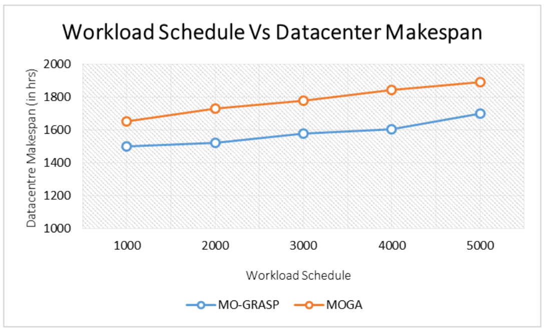

Fig. 2 depicts the comparison of datacenter makespan between proposed MO-GRASP and MOGA.One of the

Fig 2: Comparison of Datacenter Makespan

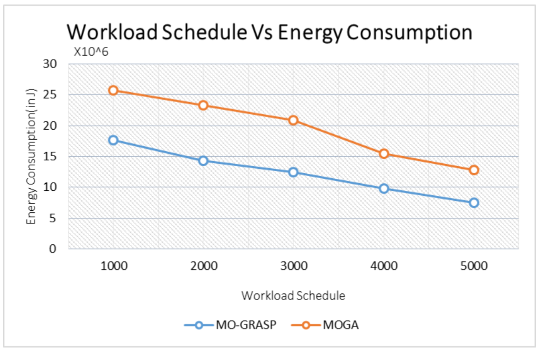

Fig 3: Comparison of Energy Consumption

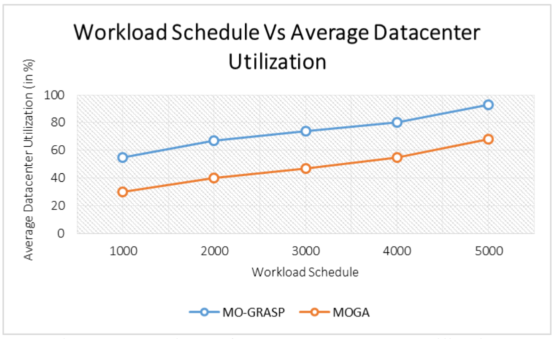

Fig 4: Comparison of Average Datacenter Utilization

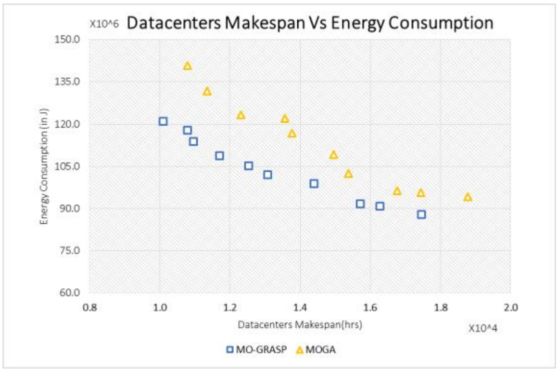

Fig 5: Non-dominatedfronts generated for Datacenter Makespan Vs Energy Consumption

main objective of this article is minimization of overall completion time of the workloads submitted to data centres (Datacenter makespan). This simulation carried out by producing five schedules.The proposed method picks an optimum datacentre for the workload requests using the objective value function defined in Algorithm 2. MO-GRASP achieves minimum makespan values compared to MOGA even though we increased the number of workloads.

Fig. 3 indicates the comparison of average datacenter utilization between proposed MO-GRASP and MOGA. The comparison results clearly shows that the proposed MO-GRASP algorithm gives better average datacentre utilization compared to MOGA. Here we found that the average utilization of the datacentre is significantly lower because of heterogeneous resources. These resources create holes during the assignment of workloads. Despite of this, MO-GRASP has shown better results as it distributes the workload among all the datacentres based on the minimum workload completion time.

Fig. 4 shows the comparison of energy consumption between proposed MO-GRASP and MOGA. The results indicate that the proposed algorithm provides optimal energy consumption as compared to the MOGA. Though MO-GRASP having higher datacenter utilization it consumes minimum energy because of threshold energy value that will be compared by the scheduler while shortlisting the datacenter for RCL.

Fig. 5 presents the comparison of datacenter makespan Vs energy consumption for proposed MO-GRASP and MOGA. It generates the non-dominated best front for both the algorithms. The evaluation consequences clearly displays that the proposed MO-GRASP algorithm gives better trade-off among makespan vs energy consumption of datacentres as compared to MOGA. The rationale behind this is, our proposed MO-GRASP algorithm uses the two level scheduler approach in which global scheduler selects the datacentre having the energy consumption below energy-threshold value and local schedulers allocate the resources at datacentre with backfilling policies.

VI. Conclusion

We have proposedMO-GRASP as a workload-scheduling algorithm for distributed datacentres. The GRASP is an adaptive random bias procedure which has been shown to run in $O(mH)$ time for m workloads and H clouds. We have presented the simulation results on workloads generated using parallel workload archive. Comparison results have shown that both the multiobjective algorithms have generated good convergence solution however proposed algorithm excels in all the parameters as compared with other multiobjective algorithm.

VII. References

1. R. Buyya, "Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility," *2009 9th IEEE/ACM Int. Symp. Clust. Comput. Grid, CCGRID 2009*, vol. 25, no. 6, p. 1, 2009.
2. K. Naik, G. M. Gandhi, and S. H. Patil, *Computational Intelligence: Theories, Applications and Future Directions - Volume II*, vol. 799. Springer Singapore, 2019.
3. http://aws.amazon.com/about-aws/global-infrastructure/Amazonn Global Infrastructure
4. http://www.microsoft.com/en-us/server-cloud/cloudos/global-datacenters.aspx.
5. https://code.google.com/p/googleclusterdata/ Google Cluster Workload Traces.
6. K. B. Naik, G. M. Gandhi, and S. H. Patil, "Pareto Based Virtual Machine Selection with Load Balancing in Cloud Data Centre," vol. 18, no. 3, pp. 23–36, 2018.
7. Y. Zhang, A. Sivasubramaniam, J. Moreira, and H. Franke, "Impact of workload and system parameters on next generation cluster scheduling mechanisms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 9, pp. 967–985, 2001.
8. T. Hagras and J. Janeček, "A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments," *Proc. Int. Conf. Parallel Process. Work.*, vol. 2003–January, no. C, pp. 149–155, 2003.
9. L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. S. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Comput. Sci. Rev.*, vol. 30, pp. 31–54, 2018.
10. X. Tao, H. Qi, W. Li, K. Li, and Y. Liu, "Profit-aware scheduling in task-level for datacenter networks," *Comput. Electr. Eng.*, vol. 61, pp. 327–338, 2017..
11. M. Moca, C. Litan, G. C. Silaghi, and G. Fedak, "Multi-criteria and satisfaction oriented scheduling for hybrid distributed computing infrastructures," *Futur. Gener. Comput. Syst.*, vol. 55, pp. 428–443, 2016.
12. S. Iturriaga, and A. Tchernykh, "Multiobjective Energy-Aware Workflow Scheduling in Distributed Datacenters", High Performance Computer Applications, vol. 595, pp. 79–93, 2016.
13. M. Guzek, J. E. Pecero, B. Dorronsoro, and P. Bouvry, "Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems," *Appl. Soft Comput. J.*, vol. 24, pp. 432–446, 2014.
14. R. K. Jena, "Multi Objective Task Scheduling in Cloud Environment Using Nested PSO Framework," Procedia Comput. Sci., vol. 57, pp. 1219–1227, 2015.
15. F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 257–271, 2018.
16. M.G.C. Resende and C.C. Ribeiro, "Greedy randomized adaptive search procedures". In J.I. Potvin and M. Gendrau, editors, Handbook of Metaheuristics, - 2nd edition, - Kluwer Academic Publishers, 2010.
17. R. Marti, V. Campos, M. Resende, and A. Duarte, "Multi-objective grasp with path-relinking," pp. 1–22, 2011.
18. http://www.cs.huji.ac.il/labs/parallel/workload/models.html
19. C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization," Fifth Int. Conf. Genet. Algorithms, vol. 93, no. July, pp. 416–423, 1993.