# Exploiting a Synergy between Greedy Approach and NSGA for Scheduling in Computing Clusters

A s m Rizvi[1], Tarik Reza Toha[2], Siddhartha Shankar Das[3], Sriram Chellappan[4], and A. B. M. Alim Al Islam [5]
[1] Department of Computer Science, University of Southern California, USA
[1, 2, 3, 5]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[4] Department of Computer Science and Engineering , University of South Florida, USA
Email: asmrizvi@usc.edu, 1205082.trt@ugrad.cse.buet.ac.bd, siddhartha047@cse.buet.ac.bd, sriramc@usf.edu, and alim_razi@cse.buet.ac.bd

***Abstract*—Computing clusters are evaluated using different performance metrics, which often appear to be conflicting while being attempted to be optimized. For such conflicting cases along with frequently having an existence of heterogeneous environment, it is difficult for the cluster administrators to efficiently schedule machines, i.e., to select the right number and right combination of machines. In this paper, we develop a technique through which cluster administrators can select the right set of machines to enhance energy efficiency and cluster performance. To do so, first, we perform extensive laboratory experiments for a period of more than one year. Based on empirical analyses of data collected from the experiments, we formulate a many-objective optimization problem for clusters and integrate a greedy approach with Non-dominated Sorting Genetic Algorithm (NSGA-III) to solve this problem. We demonstrate that our approach mostly performs better than existing approaches in the literature through both real experimentation and simulation.**

## I. Introduction

Computing clusters are extensively used for distributed and parallel computing now-a-days [24]. In a computing cluster, multiple machines work together to increase overall capacity. Fig. 1 shows the skeleton of a computing cluster. Here, a central machine normally controls other machines. The central machine acts as a master machine that distributes a big task among the several slave machines. The slave machines work together and send their results to the master machine. The master machine generally maintains coordination among the slave machines and accumulate all the results. We can use different tools for the purpose of job distribution. Hadoop [23] and Yarn [22] are examples of such kind of tools. A sophisticated mechanism such as MapReduce [4] normally runs to handle these distribution of tasks over the slave machines.

There exists several performance metrics for computing clusters [19] [21] in the literature. Here, some administrators may want to reduce the computation energy, others may want to reduce cooling energy, and so on. In this paper, we consider both the computation energy and cooling energy in combination. Besides, resource utilization and computation time are also important for some administrators. Therefore, we consider CPU usage and memory usage for resource utilization along with considering computation time.

978-1-7281-1325-8/18/$31.00 

In many cases, these objectives appear to be conflicting as well. From our year-long collected data, we find that increasing the number of machines can reduce computation time. It can also decrease total energy consumption. However, increasing the number of machines may decrease resource utilization, and increase maintenance cost as well.

Solving such conflicting objectives is a classical problem and there exist myriad techniques in the literature available to solve multiple conflicting objectives [5][6]. However, little effort has been spent to solve conflicting objectives in computing clusters. Therefore, in this paper, we propose a new approach to solve the conflicting objectives of a computing cluster. Here, we consider cooling energy consumption. The cooling energy can be around 39% of total energy consumption of a cluster [15], however, it is mostly ignored by existing studies. In road to devising our solution incorporating cooling energy, we use our year-long data to formulate a many-objective optimization problem consisting objectives and constraints of clusters. We make empirical analyses over the data to facilitate solving the optimization problem. Our solution approach exploits a synergy between greedy method and NSGA-III algorithm [5]. As the final output, our solution gives a set of machines that a cluster administrator needs to operate. We evaluate performance of our solution through both simulation and real implementation. We find that our approach performs better than other existing approaches in computing clusters. In short, we make the following contributions:

- We formulate a new many-objective optimization problem for computing clusters considering different objectives and constraints. The objectives include cooling energy consumption, which is mostly ignored in contemporary studies. Besides, we consider our year-long collected data in formulating the problem.
- We formulate a new approach exploiting both greedy method and NSGA-III algorithm to solve the many-objective optimization problem for computing clusters. Our technique pinpoints the set of machines that need to be selected to achieve the cluster objectives.
- We use a well-established simulation platform namely *SimGrid* to experimentally evaluate performances of our proposed and other existing approaches.
- We also implement our proposed and existing ap-

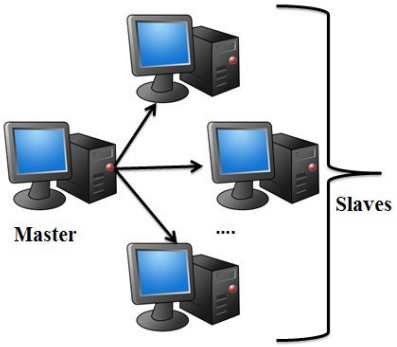



Fig. 1: Master-slave relation in a computing cluster

proaches in a real setup and evaluate their performance. Comparative analysis over all the experimental results demonstrates that our proposed approach can provide significant performance improvement in most of the cases compared to other existing approaches.

## II. Related Work

There are several studies in the literature, which consider performance optimization in cluster computing. For example, the study in [21] presents a stochastic approach for the performance optimization. This study does not consider any many-objective optimization approach for more than three objectives [3], rather it uses a multi objective stochastic approach. Moreover, it does not consider optimizing cooling energy consumption.

Besides, multi-objective optimization techniques are studied in virtual machine based schemes [11]. There are some other studies [14], which consider resource provisioning techniques in cloud computing. However, these studies do not use any specialized many-objective optimization technique. At the same time, they do not integrate any empirical performance characterization of clusters.

Some recent studies focus on multi-objective performance optimization in computing clusters and clouds. Examples include a Particle Swarm Optimization (PSO) based approach [12] and Ant Colony Optimization (ACO) based approach [8]. These techniques are yet to be extended for many-objective cases. Moreover, integrating empirical performance characterization of clusters is yet to be focused by all the approaches in the literature to the best of our knowledge. Such empirical characterization exhibits a potential to reveal impacts of operational factor over cluster performance - yet another aspect to be focused in the literature.

## III. Impacts of Operational Factors over Performance of a Cluster

We analyze impacts of operational factors over cluster objectives from our experimental data. Here, we consider the number of machines, configuration of the machines, machine failures etc, as the operational factors.Besides, we consider four cluster objectives, which we elaborate next.

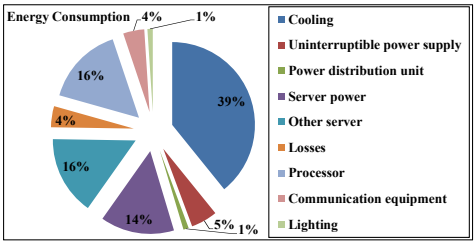



Fig. 2: Energy consumption of data centers in USA [15]

### A. Cluster Objectives

We define the four cluster objectives in the following way: **Objective 1 - Minimizing computation time:** Computation time means the time to finish a task. Cluster administrators generally want to decrease the computation time so that tasks are finished within a shortest possible duration. From Fig. 3a (got from our year-long experiment with settings as presented in Section VII-A), we can see that when the number of machines increases, the computation time decreases. We make our experiment in different seasons of the year and find the same pattern. Besides, if we increase the workload, the computation time also gets increased.

**Objective 2 - Minimizing total energy consumption:** In computing clusters, we want to decrease total energy consumption. Fig. 2 suggests that cooling energy plays a significant role in total energy consumption ($\sim$39%). Hence, we incorporate cooling energy while considering total energy consumption. From Fig. 3b, we can see a trend line of decreasing energy consumption with an increase in the number of machines. This behavior seems surprising. We get energy by multiplying power and computation time. Computation time is decreased with the increase of number of machines. Total power consumption is increased with the increase of number of machines. However, the ratio of decreasing computation time is much greater than the increase in power consumption. That is why we get a decrease in energy consumption.

**Objective 3 - Minimizing cost:** Cost to operate a computing cluster increases with an increase in the number of machines. With the increase in the number of machines, other related costs such as that for maintenance, utilities etc. will also get increased. Cluster administrators want to decrease the cost of a cluster.

**Objective 4 - Maximizing utilization:** Administrators also want to improve the resource utilization. When a machine runs, it consumes resources such as physical memory and CPU. Getting higher CPU and memory utilizations is more desirable than smaller utilization. From Fig. 3c, we can see CPU utilization decreases with an increase in the number of machines.

In the following subsection, we discuss impacts of different operational factors on cluster objectives.

### B. Impacts of Different Factors on Cluster Objectives

Different operational factors have significant impacts over cluster objectives. Here, we discuss how these factors have impacts on cluster objectives.

Impacts of the number of machines: Cluster objectives are dependent on the number of operating machines. They have conflicting impacts over cluster objectives. By increasing the number of machines, we can get decreased computation time. At the same time, the total energy consumption gets decreased. We can achieve these two objectives with an increased number of machines. However, resource utilization gets decreased with an increase in the number of machines. As more machines are now working, they will not be utilized according to their capacities. At the same time, the operating cost also gets increased, which is not desirable. Hence, the number of machines have conflicting impacts over cluster objectives. Fig. 3 shows the impacts of increasing number of machines.

Impacts of configuration of computing machines: Configuration of machines have impacts on cluster objectives. High-performing machines can reduce computation time. At the same time, these machines can increase energy consumption and overall cluster cost.

## IV. Proposed Problem Formulation

We define configuration of a machine as the machine property. We can have different machine properties such as physical memory, processor speed, etc. We represent these properties as $P_1, P_2, P_3, ..., P_N$, where $N$ is the number of machine properties. In our experiment, we take network bandwidth, memory usage, and CPU usage as machine properties. We express effectiveness of a machine using the term Machine Value, $MV$, which can span over [0, 100]. We define $MV$ of a machine $i$ by $MV_i = w_1 \times P_1 + w_2 \times P_2 + w_3 \times P_3 + ... + w_N \times P_N$, where $w_1, w_2, w_3, ..., w_N$ are the weights of $N$ properties. Based on these terms, we develop our problem formulation considering the four objectives as presented in the earlier section.

**Minimizing computation time:** Computation time is dependent on the number of machines and machine value. As presented earlier in Fig. 3a, it decreases with an increase in the number of machines. Computation time also decreases when the machine value gets increased (using high-performing machines). Thus, from the perspective of machine value, we deduce our objective as follows:

$$\max \quad \sum_{i=1}^{N_M} s_i (MV)_i \tag{1}$$

Here, $N_M$ is the total number of machines in the cluster. $s_i$ is a decision variable that describes an indicator function. $s_i$ is 1 if we run $i^{th}$ machine operational in our cluster and 0 if we do not make $i^{th}$ machine operational. We can write $s_i$ as

$$s_i = \begin{cases} 1, & \text{if machine } i \text{ is operating in the cluster} \\ 0, & \text{otherwise} \end{cases}$$

To remain coherent with other objectives, we make this optimization problem as a minimization problem and make necessary changes. Thus, the objective function becomes:

$$min \quad \sum_{i=1}^{N_M} (100 - s_i (MV)_i) \tag{2}$$

Note that, computation time also depends on work load. If the workload gets increased, the computation time gets increased. Considering the workload for machine $i$ to be $W_i$, we adopt the following normalized values for the objective function:

$$\min \quad \frac{\sum_{i=1}^{N_M} (100 - s_i (MV)_i)}{\sum_{i=1}^{N_M} s_i \times 100} \times WT_1 + \frac{W_i}{W_{max}(i)} \times WT_2 \tag{3}$$

Here, $W_{max(i)}$ refers to the maximum allowable workload for machine $i$. Besides, $W_i$ is the machine workload, which we assume to be evenly balanced as $\frac{Total_{workload}}{\sum_{i=1}^{N_M} s_i}$. $WT_1$ and $WT_2$ are the weights for machine configuration and workload. In our analysis, we adopt same valued weights in both parts of Eq. 3. In case we would give less weight to machine configuration, some highly-configured machines might not be selected with high probability.

**Minimizing total energy consumption:** Energy consumption is dependent on the number of machines and temperature difference. From Fig. 3b, we can see that total energy consumption exhibits a decreasing trend with an increase in the number of machines. Besides, if the temperature difference ($T_{diff}$) decreases, the total energy consumption gets decreased. In our model, there is a range for allowable temperature. Let $T_E$ is the environment temperature, $T_M$ is the maximum allowable temperature within the cluster, $T_L$ is the lowest value of allowable temperature range, and $T_D$ is an expected temperature that can be within $T_L$ and $T_M$. Thus, we can deduce as: $T_{diff} = T_E - T_D$. The cooling system needs to cool the system by $T_{diff}$. Here, the maximum temperature difference can be $T_{MaxDiff} = T_E - T_L$. Thus, we deduce the following objective function:

$$\begin{aligned} \min \quad & \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times WT_3 + \frac{T_{diff}}{T_{MaxDiff}} \times WT_4 \\ \text{subject to} \quad & T_L \leq T_D \leq T_M \end{aligned} \tag{4}$$

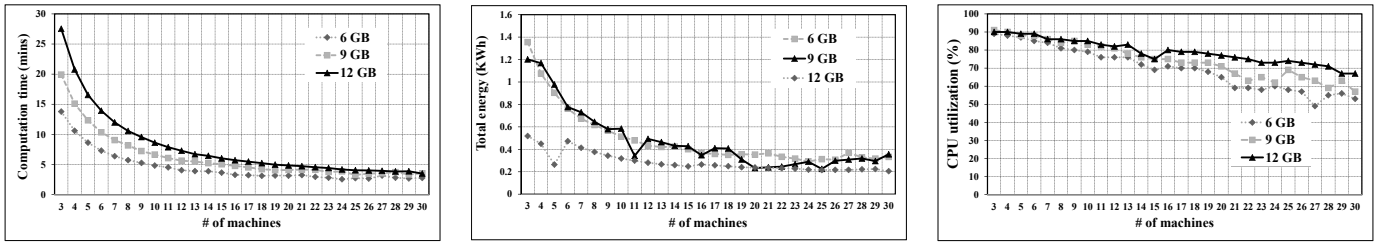
Here, $WT_3$ and $WT_4$ indicate weights for the number of machines and temperature difference. In our analysis, we assume same values for the weights in Eq. 4. We can use different weights if we intend different effects of number of machines, workload, and temperature difference. Due to space limitation we cannot provide details in selecting weights.

**Minimizing cost:** Cost of a cluster is dependent on the number of operating machines and their configuration. Cost will be increased with an increase in the number. Besides, high performing machines will also increase the cost. Therefore, we deduce an objective function as follows:

$$\min \quad \sum_{i=1}^{N_M} s_i (MV)_i \tag{5}$$

**Maximizing utilization:** As already shown in Fig. 3c, utilization increases if we decrease the number of machines. Therefore, we deduce an objective function as:

$$\min \quad \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times 100 \tag{6}$$




(a) Computation time comparison (b) Total energy consumption comparison (c) CPU utilization comparison

Fig. 3: Comparative analysis of cluster objectives with an increase of number of machines

Note that, if there are $\sum_{i=1}^{N_M} s_i$ number of operational machines and $W_i$ workload, then there should be $\frac{W_i}{\sum_{i=1}^{N_M} s_i}$ workload per machine on an average. Considering $HDD_i$ as the size of available hard disk of machine $i$ , we can write the constraint as $\forall i \frac{W_i}{\sum_{i=1}^{N_M} s_i} \leq HDD_i$. Combining all the above objectives and constraints we can write:

$$min \begin{cases} \frac{\sum_{i=1}^{N_M} (100 - s_i(MV)_i)}{\sum_{i=1}^{N_M} s_i \times 100} \times WT_1 + \frac{W_i}{W_{max(i)}} \times WT_2 \\ \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times WT_3 + \frac{T_{diff}}{T_{MaxDiff}} \times WT_4 \\ \sum_{i=1}^{N_M} s_i(MV)_i \\ \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times 100 \end{cases}$$

$$subject\ to \begin{cases} \forall i \frac{W_i}{\sum_{i=1}^{N_M} s_i} \leq HDD_i \\ T_L \leq T_D \leq T_M \\ \sum_{i=1}^{N_M} s_i(MV)_i > 0 \end{cases}$$

Next, we present a solution for optimizing the above optimization functions.

## V. Proposed Solution Approach

Our focus covers a many-objective optimization problem. There are many approaches to solve such problems, which can be found in [3]. In our case, we use NSGA-III [5] as our base algorithm. We exploit outcomes of our empirical analyses to modify the NSGA-III algorithm. Several functions usually participate in the original NSGA-III optimization algorithm such as selection, crossover, and mutation. We use our empirical results to design these functions. Here, we consider selection decision of a machine as a decision variable. If there are $N_M$ machines, then we will have $N_M$ number of decision variables. We also consider the expected temperature (expressed as $T_D$) as a decision variable. This expected temperature will be maintained by the cooling devices. Hence, our algorithm will finally give the number of machines to be active or operational in the cluster, a selected set of machines, and a temperature that needs to be maintained by the cooling devices. Now, we are describing different modifications over the functions which are actively used in optimization steps.

TABLE I: Impacts of selection threshold value in experimental results with 120 pareto front solutions

| Threshold value | No. of solutions with $< 6$ machines | Computation time (mins) | Cooling energy (KWh) | Computation energy (KWh) |
|---|---|---|---|---|
| 2 | 34 | 11.1 | 0.43 | 0.92 |
| 4 | 27 | 9.5 | 0.31 | 0.66 |
| 6 | 13 | 9.7 | 0.24 | 0.53 |
| 8 | 14 | 10.2 | 0.32 | 0.68 |

### A. Population Selection

From our experiment, we find that if the number of machines is below than a particular number, then both computation time and total energy exhibit highly increased values. Besides, Fig. 3a and 3b, we can see that the rate of decrement in computation time and total energy is very high when the number of machines is less than 6. After this range, the changing rate gets significantly decreased. Based on this observation, while we select a population for the next generation, we eliminate the population that has fewer operating machines. In our case, we take this threshold value, $T_h$ as $\frac{N_M}{6}$.

From Table I, we can see when the threshold value is 2, we have undesirable values for computation time, cooling energy, and computation energy. When we select small threshold value we have more solutions with small number of machines among 120 pareto front solutions. These solutions practically increase the chance to cause high computation time and energy. We use jMetal [7] as the objective optimization framework and modify its existing selection, mutation and crossover functions. Algorithm 1 shows the steps for modified population selection.

**Algorithm 1** Population Selection

```
1: function SELECTPOPULATION(Threshold value, T_h)
2:     G ← Existing generation
3:     for each population p ∈ G do
4:         populationSize ← populationSize(p)
5:         if populationSize > T_h then
6:             newGeneration.add(p)
7:     Fill up the generation with random population
8:     return newGeneration
```

### B. Crossover

While making crossover, we use a partition to separate machine selection variables and temperature variable as they

are two different types of variables. We make crossover among machine selection variables while not mixing temperature variable with the selection variable. Algorithm 2 represents the modified algorithm for crossover function. We make three steps for our modification. Here, following a greedy approach, we introduce biases for the highly-performing machines while making crossover. This ensures higher chances of their selection in a next generation. We describe the three steps crossover based on the bias below:

**Half uniform crossover:** We apply half-uniform crossover [9][20] as our primary crossover approach. As we have binary decision variable, we only make crossover when the particular chromosome is different from each other. We can see the half uniform crossover in line 11.

**Crossover based on clustering:** We separate the $N_M$ number of machines into two clusters for each property. Line 5 of Algorithm 2 shows this. We use Euclidean distance in k-means clustering [13] to make two separate clusters. We deploy Cluster 3.0 [1] tool for clustering. We name these two clusters as top performing and bottom performing clusters. We make a set ($S_T$) of machines that are in the top-performing cluster for all properties and a set of machines ($S_B$) that are in the bottom-performing cluster for all properties. Line 6 and 7 of Algorithm 2 show this. From $S_T$, with some probabilistic condition, we take machines into our next generation. Besides, from $S_B$, with some probabilistic condition, we do not take that machine into our next generation. We can see this in Line 12-17 of Algorithm 2.

**Take the top and remove the bottom:** With some probabilistic condition, we take the topmost machine for one property, which is also selected in a stochastic manner. Besides, we also exclude the bottom most performing machine for the same property. Line 19-22 of Algorithm 2 present these modifications.

### C. Solution Filtering

Optimization methods give a pareto-front with multiple solutions. Solutions having a desirable value for one objective sometimes have an undesirable value for other objectives. These solutions are not acceptable since they make significant performance degradation for some objectives. Hence, we take only those solutions, which have values from 25% to 75% of all values pertinent to all objectives. Among these solutions, based on the administrator's defined weight function, we select only one solution. We use the following function to converge four objective values into one measure:

$$F_{total} = W_{obj1} \times V_{obj1} + W_{obj2} \times V_{obj2} + \cdots + W_{objN} \times V_{objN} \quad (7)$$

In Eq. 7, every objective value, $V_{obj}$ should be within 25% to 75% of the corresponding objective value to be considered as a feasible solution. Weights of these objectives will be defined by the cluster administrator. After having the feasible solutions, we sort over the $feasibleSolution$. Then, we take the top solution having the lowest merged objective values since we convert our problem into a minimization problem. Algorithm 3 shows the filtering process. Here, $firstBoxPlot$ refers to 25% and $thirdBoxPlot$ refers to 75% of corresponding objective value.

**Algorithm 2** Population Crossover

1: **function** CROSSOVER(Parent $P_1$, Parent $P_2$)
2: $C_1 \leftarrow$ Chromosome set in $P_1$
3: $C_2 \leftarrow$ Chromosome set in $P_2$
4: $noOfCluster \leftarrow 2$
5: Implement k-means clustering for all machine properties with $noOfCluster$
6: Find out the machines, which are in the top group for all the machine properties, $S_T$
7: Find out the machines, which are in the bottom group for all the machine properties, $S_B$
8: $size \leftarrow C_1.size() - 1$
9: $i = 0$
10: **while** $i < size$ **do**
11: Inter-change $i^{th}$ chromosome value of $C_1$ and $C_2$ based on probability and if they are different
12: **if** *$i$ is in $S_T$ and Random.nextDouble() < insertProbability* **then**
13: $C_1(i).set(1)$
14: $C_2(i).set(1)$
15: **if** *$i$ is in $S_B$ and Random.nextDouble() < deletionProbability* **then**
16: $C_1(i).set(0)$
17: $C_2(i).set(0)$
18: $i++$
19: **if** *Random.nextDouble() < takeTopProbability* **then**
20: $rndProperty$ = Random.nextInt() mod $noOfproperty$
21: Take the top machine for $rndProperty$
22: Remove the bottom machine for $rndProperty$ property
23: Update $P_1$ and $P_2$ according to $C_1$ and $C_2$

**Algorithm 3** Solution Filtering

1: **function** SOLUTIONFILTERING(objectiveValues, O)
2: **if** for all objectives $o.val() \geq O.firstBoxPlot()$ and $o.val() \leq O.thirdBoxPlot()$ **then**
3: $feasibleSolution.add(O)$
4: Sort $feasibleSolution$ in increasing order
5: return $feasibleSolution.get(0)$

## VI. Simulation Evaluation

We evaluate performances of our algorithm and other existing approaches in a simulation environment. We adopt clusters having 30 and 50 machines to evaluate our algorithm and compare it with other approaches.

### A. Simulation Platform and Settings

For simulation, we use a well-known simulation tool for distributed system called *SimGrid* [2].

*SimGrid* has an energy plug-in, which can compute both computation energy and cooling energy [17]. We use this plug-in in our simulation environment, which is shown in Table II. Note that *SimGrid* uses different unit conventions while setting-up machines in clusters. The conversions between *SimGrid* unit and the traditional unit can be found in [17], which we adopt in our case.

TABLE II: Simulation environment in SimGrid

| Parameter | Value |
|---|---|
| *SimGrid* version | 3.12 |
| # of master machine | 1 |
| # of slave machines | 29 and 49 |
| PC power | 4,000-38,000 Mega FLOPS |
| PC power consumption | Peak: 10 - 400 W,<br>idle: 2.5-100 W,<br>power off: 5 W |
| Line bandwidth | 10-100 KBps |
| Total # of files | 86, 64 and 36 |
| Size of each file | 787 MB |
| Total data size | 67.7 GB, 50.4 GB and 28.3 GB |
| Maximum allowable temperature | $20^\circ - 23^\circ$ C |
| Environment temperature | $28^\circ$ C |

| Workload | Improvement over PSO (%) | | | Improvement over ACO (%) | | |
|---|---|---|---|---|---|---|
| | Time | Cooling energy | Comp. energy | Time | Cooling energy | Comp. energy |
| 67.7 GB | 21 | 13 | 10 | 43 | 10 | 5 |
| 50.4 GB | 36 | 11 | 10 | 17 | 8 | 8 |
| 28.3 GB | 43 | 13 | 10 | 15 | 5 | 0 |

TABLE III: Improvement over PSO and ACO for various workloads in *SimGrid* with 30 machines

### B. Simulation Results

Fig. 4 shows simulation results in *SimGrid* for 30 machines and Fig. 5 shows the simulation results for 50 machines. Table III and IV show corresponding improvements achieved by our approach for 30 and 50 machines respectively.

### C. Findings from Simulation Results

We find from Table III and IV that modified NSGA-III gives better performance than other methods for all three objectives in most cases. Here, Table III suggests that computation time gets improved by 21%, 36%, and 43% than PSO, and 43%, 17%, and 15% than ACO in a cluster of 30 machines having different workloads. Besides, cooling energy gets improved by 13%, 11%, and 13.0% compared to PSO, and by 10%, 8%, and 5% compared to ACO. Computation energy also gets improved by the modified NSGA-III algorithm. Here, we have nearly 10% improvement over PSO and nearly 5% to 8% improvement over ACO. We also find similar or even better performance in most of the cases for 50 machines as shown in Table IV.

## VII. Evaluation through Real Implementation

We create a real cluster, and evaluate performance of our approach and other approaches with different workloads. We present settings of our implementation and its experimental results below.

| Workload | Improvement over PSO (%) | | | Improvement over ACO (%) | | |
|---|---|---|---|---|---|---|
| | Time | Cooling energy | Comp. energy | Time | Cooling energy | Comp. energy |
| 67.7 GB | 38 | 59 | 59 | 16 | 55 | 55 |
| 50.4 GB | 33 | 36 | 38 | 41 | 79 | 79 |
| 28.3 GB | 10 | -6 | 0 | 46 | 87 | 87 |

TABLE IV: Improvement over PSO and ACO for various workloads in *SimGrid* with 50 machines

TABLE V: Experimental settings in real implementation

| Parameter | Value |
|---|---|
| # of master machine | 1 |
| # of slave machines | 29 |
| Processor | Intel Core 2 Duo |
| Processor base frequency | 2.4 GHz (3 PCs), 2.66 GHz (10 PCs), 2.8 GHz (17 PCs) |
| Memory | 1 GB to 2 GB |
| Network B/W | 5 to 100 MBps |
| OS | Ubuntu 14.04 LTS (x86) |
| Total number of files | 40, 24, 16, 11 |
| Size of each file | 787 MB |
| Total data size | 31.5 GB, 18.9 GB, 12.6 GB, and 8.7 GB |

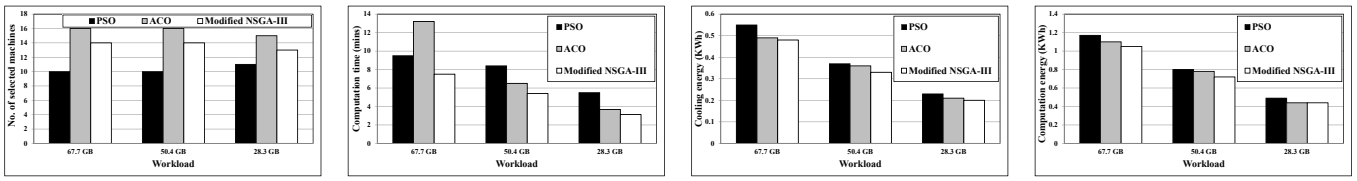
### A. Settings of Real Implementation

We implement a real cluster having thirty machines. Among these machines, one is selected as the master node and the others as slave nodes. Distribution of tasks from the master node to slave nodes is shown in Fig. 1. We use Hadoop [23] framework to implement distributed system in the cluster. We set up a distributed and multi-node (30 PCs) Apache Hadoop cluster with Hadoop Distributed File System (HDFS), running on Ubuntu Linux [16]. We vary the work-load and use a different number of machines to get the computation energy and cooling energy consumptions. Here, we run the popular word-count task [10]. Table V shows the experimental setup in more detail. Beside, Fig. 7 shows a snapshot of real implementation.

We use two Arduino energy monitors to get the computation energy and cooling energy. One monitor measures the CPU power and the other measures the air condition power. We use potential transformer (PT) and current transformer (CT) to measure power and current consumed by the cluster machines and cooling devices. Using the basic power formula, we get computation power and cooling power. Power consumed by air conditions is considered as the cooling power. After that, using $Work = P \times T$ formula, we get the computation energy and cooling energy where $P$ refers to power and $T$ refers to time. One thing is to be noted that, computation power means the power, necessary to finish the given task by the slave nodes. We test our approach in a heterogeneous environment to see the effect of our greedy approach. To create heterogeneity, we use *Wondershaper* tool [18] to control the network bandwidth. *SimGrid* and real testbed use different unit conventions. Hence, imitating the real testbed into *SimGrid* is not always completely possible. Besides, we use different performance metrics for real testbed and simulation environment. For example, in real setup, we can get resource utilization, however, in the existing *SimGrid* we cannot get such resource utilization.

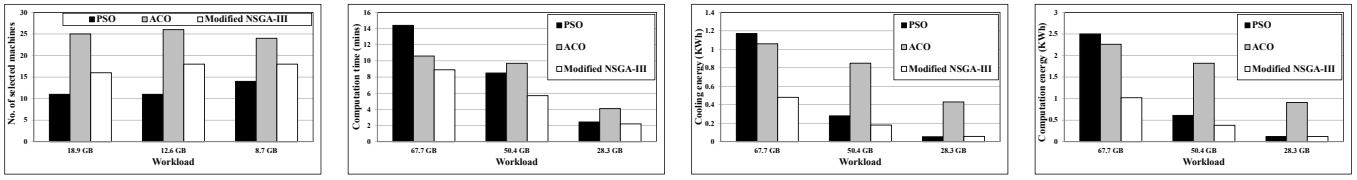
### B. Results from Real Implementation

We use different workloads and consider five cluster objectives. We show all comparisons against other approaches in Table VI. We have up-to 74% improvement in computation time, 75% improvement in cooling energy, 63% improvement in computation energy, 127% improvement in CPU usage, and 5% improvement in memory usage over PSO. Moreover, we have up-to 49% improvement in computation time, 49% improvement in cooling energy, 58% improvement in computation energy, 159% improvement in CPU usage, and 4% improvement in memory usage over ACO.



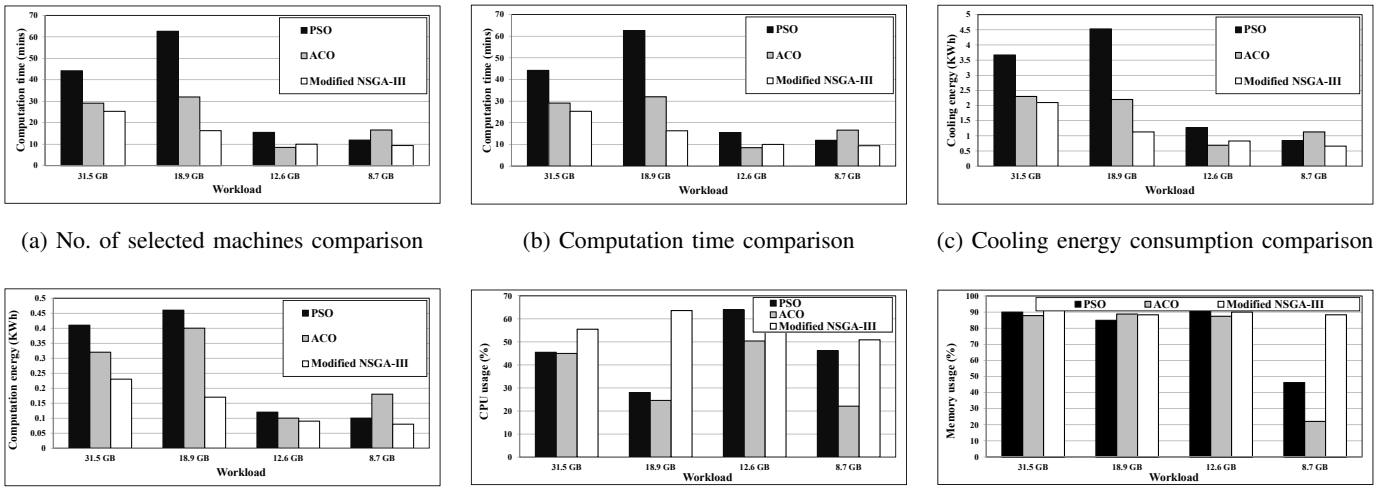

(a) No. of selected machines (b) Computation time (c) Cooling energy consumption (d) Computation energy consumption

Fig. 4: Comparison of different algorithms with various workloads in *SimGrid* for a cluster with 30 machines cluster




(a) No. of selected machines (b) Computation time (c) Cooling energy consumption (d) Computation energy consumption

Fig. 5: Comparison of different algorithms with various workloads in *SimGrid* for a cluster with 50 machines cluster




(a) No. of selected machines comparison (b) Computation time comparison (c) Cooling energy consumption comparison

(d) Computation energy consumption comparison (e) CPU usage comparison (f) Memory usage comparison

Fig. 6: Comparison over performances of different algorithms with various workloads in a real cluster with 30 machines

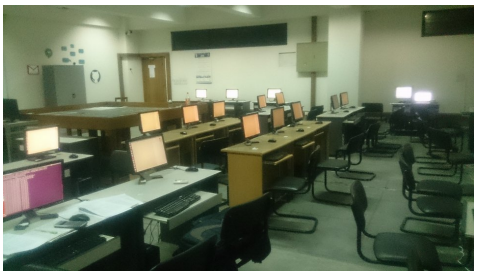


Fig. 7: Snapshot of real implementation

### C. Findings from Results of Real Implementation

We compare our algorithm and other existing algorithms in the real testbed as mentioned in Subsection VII-A. We take the number of machines, computation time, computation energy, cooling energy, CPU usage, and memory usage as our cluster objectives. We want to decrease the number of machines (to decrease cost), computation time, computation energy and cooling energy. At the same time, we want to increase the CPU and memory usage. From Table VI, we can see the comparison. For the case of 31.5 GB and 18.9 GB workload, NSGA-III shows significant performance improvement than

TABLE VI: Improvement over PSO and ACO with various workloads in a real testbed

| Workload | Improvement over PSO (%) | | | | | Improvement over ant colony optimization (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Cooling energy | Computation energy | CPU usage | Memory usage | Time | Cooling energy | Computation energy | CPU usage | Memory usage |
| 31.5 GB | 43 | 75 | 22 | 1 | 5 | 13 | 49 | 28 | 23 | 4 |
| 18.9 GB | 74 | 75 | 63 | 127 | 5 | 49 | 49 | 58 | 159 | 0 |
| 12.6 GB | 36 | 35 | 25 | 0 | -1 | -18 | -20 | 10 | 27 | 3 |
| 8.7 GB | 21 | 21 | 20 | 10 | -7 | 43 | 42 | 56 | 130 | 1 |

the other two approaches for all the objectives. For 12.6 GB workload, NGSA-III performs better in terms of most of the cluster objectives. For 8.7 GB workload, though PSO has a better memory usage, NSGA-III performs better for all other objectives. We have some negative values also. This is because evolutionary algorithms cannot assure optimal solutions all the time.

## VIII. Conclusion

Solving conflicting objectives in clusters to provide administrators a set of machines is important. However, little research effort has been made till now to solve this problem for computing clusters. Moreover, the few available studies focus on this from either single or multi-objective perspectives. On the contrary, in this paper, we provide a many-objective solution for cluster administrators to select the cluster nodes. Here, we exploit a synergy between a greedy approach and NSGA-III algorithm to solve the many-objective problem. We evaluate performance of our approach and other existing approaches. Comparative analysis over the performances demonstrates that our proposed approach reveals the best set of nodes in most of the cases, which fulfill all the cluster objectives. We experiment in both simulation environment and in a real cluster, which confirms efficacy of our approach. To facilitate developing our solution approach, we collect real data for a period of more than one year and conduct empirical analyses of the data. We consider different environmental settings in different seasons of the year to ensure robustness of the empirical analyses. Our approach engendered from the analyses can be used in real clusters to select the best combination of machines, which will fulfill the cluster objectives.

## Acknowledgement

This work was supported by the ICT Division, Government of the People's Republic of Bangladesh.

## References

[1] A. Bender. An evaluation of cluster 3.0 as a general tool for principal component analysis. In *Proceedings of the* $47^{th}$ *SIGCSE*, pages 725–725. ACM, 2016.

[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Simgrid: a sustained effort for the versatile simulation of large scale distributed systems. *CoRR*, abs/1309.1630, 2013.

[3] S. Chand and M. Wagner. Evolutionary many-objective optimization: a quick-start guide. *Surveys in Operations Research and Management Science*, 20(2):35–42, 2015.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[5] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601, 2014.

[6] K. Deb, A. Pratap, S. Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[7] J. J. Durillo and A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[8] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242, 2013.

[9] X. Hu and E. Di Paolo. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 55–62. IEEE, 2007.

[10] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 41–51. IEEE, 2010.

[11] R. Li, Q. Zheng, X. Li, and J. Wu. A novel multi-objective optimization scheme for rebalancing virtual machine placement. In $9^{th}$ *IEEE CLOUD*, pages 710–717, 2016.

[12] C. Lijun and L. Xiyin. Modeling server load balance in cloud clusters based on multi-objective particle swarm optimization. *IJGDC*, 8(3):87–96, 2015.

[13] A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[14] S.S. Manvi and G.K. Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424–440, 2014.

[15] Mason and Hanger. *Data Center Energy Consumption*, Dec 2015. Available online at http://www.ha-inc.com/blog/entry/data-center-energy-consumption/. Online accessed on April 1, 2017.

[16] M.G. Noll. Running Hadoop on Ubuntu Linux. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/, 2016. [Online accessed on February 22, 2016].

[17] A. Rizvi, T.R. Toha, M.M.R. Lunar, M.A. Adnan, and ABM A. Al Islam. Cooling energy integration in simgrid. In *NSysS*, pages 132–137. IEEE, 2017.

[18] Ruchi. Wodershaper Tool. http://www.ubuntugeek.com/wondershaper-simple-traffic-shaping-tool.html, 2016. [Online accessed on August 01, 2017].

[19] G. Singh, C. Kesselman, and E. Deelman. A provisioning model and its comparison with best-effort for performance-cost optimization in grids. In *Proceedings of the* $16^{th}$ *HPDC*, pages 117–126. ACM, 2007.

[20] W. M. Spears and V. Anand. A study of crossover operators in genetic programming. In *International Symposium on Methodologies for Intelligent Systems*, pages 409–418. Springer, 1991.

[21] K.M. Tarplee, A.A. Maciejewski, and H.J. Siegel. Robust performance-based resource provisioning using a steady-state model for multi-objective stochastic programming. *IEEE Transactions on Cloud Computing*, 2016.

[22] S. Wadkar, M. Siddalingaiah, and J. Venner. *Pro Apache Hadoop*. Apress, 2014.

[23] T. White. *Hadoop: The Definitive Guide*. O'Reilly, first edition edition, june 2009.

[24] K. Xiong and S. Suh. Resource provisioning in sla-based cluster computing. In *Workshop on JSSPP*, pages 1–15. Springer, 2010.